**VŠB - Technická univerzita Ostrava**
**Fakulta elektrotechniky a informatiky**
**Katedra informatiky**

# Možnosti monitorování informačních systémů pomocí systému Microsoft System Center

# Options of Monitoring Information Systems Using the System Microsoft System Center

2010 Petr Farkas

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

Student: **Bc. Petr Farkas**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Možnosti monitorování informačních systémů pomocí systému Microsoft System Center
Options of Monitoring Information Systems Using the System Microsoft System Center

Zásady pro vypracování:

Cílem této práce je seznámit se s možnostmi monitorování informačních systému pomocí Microsoft System Center (MSC). Tento produkt představuje ucelené řešení pro správu umožňující IT specialistům v organizacích střední velikosti proaktivně a efektivně spravovat jejich IT prostředí.

Samotná práce se skládá z následujících bodů:
1. Nastudování produktu Microsoft System Center (MSC).
2. Vytvoření testovací aplikace, která bude simulovat reálnou aplikaci automatického zpracování smyšleného procesu.
3. Vytvoření tzv. management package, který bude aplikaci monitorovat.
4. Vytvoření dokumentace popisující standardní scénáře nasazení MSC a podmínky jeho realizace.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Tomáš Kocyan**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010


doc. Dr.Ing. Eduard Sojka
*vedoucí katedry*

prof. Ing. Ivo Vondrák, CSc.
*děkan fakulty*

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě dne 30. 4. 2010 …….....

# Abstrakt

Diplomová práce se zabývá možnostmi monitorování informačních systémů pomocí System Center Essentials. Nejprve jsou rozebrány dva populární nástroje sloužící k usnadnění zápisů zpráv. Jedná se o projekt log4net a řešení od společnosti Microsoft, které je součástí Enterprise Library jako jeden samostatný blok nazvaný Logging Application Block. Na tuto část navazuje pojednání o samotném produktu System Center Essentials a jeho vlastnostech. Největší důraz je ale kladen na vytváření řídícího balíčku. Řídící balíček umožňuje nakonfigurovat Systém Center Essentials na míru konkrétnímu informačnímu systému nebo aplikaci.



# Klíčová slova

Logování, Microsoft Enterprise library, Systém Center Essentials, Log4Net**,** Management pack, Logging Application Block.



# Abstract

The thesis deals with the possibilities of information systems monitoring using the System Center Essentials. Two popular frameworks are discussed, as the tools to facilitate the log entry. It is the open source project log4net and the Logging Application Block, which is a separate part of Microsoft Enterprise Library. In the next part of the thesis, the product System Center Essentials itself is described. The greatest emphasis is placed on creating a Management Package. The Management Pack allows configure System Center Essentials tailored to a specific information system or application.



# Keywords

Logging, Microsoft Enterprise library, System Center Essentials, Log4Net, Management pack, Logging Application Block

# Seznam použitých zkratek:

CPU……..…… Central Processing Unit
EXE……..... Executable File
HTML…….HyperText Markup Language
IT……..……Informační technologie
LDAP……..……Lightweight Directory Access Protocol
MAML…….……...Microsoft Assistance Markup Language
MSI…….……Microsoft Windows Installer
OLE DB……..……Object Linking and Embedding Database
SCE…….……Systém Center Essentials
SDK……Software development kit
SMS……Short Message Service
SNMP…….....Simple Network Management Protocol
W3C……World Wide Web Consortium
WCF……... Windows Communication Foundation
WMI……... Windows Management Instrumentation
WPF……Windows Presentation Foundation
WSUS…….……Windows Server Update Services
WWW ……World Wide Web
XML……... Extensible Markup Language

# Obsah

# 1 Úvod

V současné době jsou kladeny stále vyšší nároky na kvalitu a především stabilitu informačních systémů. Většina uživatelů vyžaduje, aby systém běžel sedm dní v týdnu, čtyřiadvacet hodin denně. Proto je důležité mít stálý dohled nad systémem. Jedním z nástrojů, který umožňuje proaktivní monitorování, je System Center Essentials od firmy Microsoft.

Cílem této práce je seznámit se s možnostmi monitorování informačních systému pomocí System Center Essentials. Produkt System Center představuje ucelené řešení pro správu umožňující IT specialistům v organizacích střední velikosti proaktivně a efektivně spravovat jejich IT prostředí. Praktickou součástí diplomové práce je vytvoření testovací aplikace, umožňující simulovat reálnou aplikaci automatického zpracování smyšleného procesu. Je nutné, aby aplikace umožňovala používat více typu logování, a také nastavení logovacího formátu, úrovně, výstupu atd. Pro tuto aplikaci je vytvořen řídící balíček, tzv. management package, pro System Center Essentials, který ji bude monitorovat. Praktická část práce má za cíl ověřit získané teoretické znalosti z této oblasti v praxi.

První část práce popisuje možnosti vytváření a zápisu logů v aplikacích. Tato část se skládá ze tří kapitol. V první jsou rozebrány rozdíly mezi logováním a trasováním. Další dvě kapitoly popisují logovací nástroje Log4net a Logging Application Block z Enterprise Library. Při popisu je kladen důraz na nastavení konfiguračního souboru včetně ukázek kódu a popisů jednotlivých elementů.

Druhá část práce je věnována problematice monitorování softwaru a hardwaru pomocí System Center Essentials. Jsou zde uvedeny mimo jiné rozdíly oproti předchozí verzi, scénáře nasazení nebo také architektura tohoto nástroje. Důraz je kladen především na tvorbu řídícího balíčku pomocí jazyka XML a jeho strukturu.

# 2 Trasování a logování

Během vývoje i následného provozování aplikací je potřeba sledovat chování aplikace a získávat nějakým způsobem informace o událostech, které v běžícím programu nastanou. Ke sledování informací při vývoji aplikace slouží trasování. Zapsané informace se nazývají logy. Získané informace jsou využity k diagnostice problémů s programem a k ladícím účelům.

Logování událostí poskytuje informace pro diagnostiku správci systému. Událostí je každá významná situace, která nastane v průběhu života aplikace a vyžaduje pozornost uživatele nebo záznam do protokolu. Z protokolu pak lze zjišťovat informace o systému nebo sledovat zabezpečení a to jak na místním, tak na vzdáleném počítači.

### Trasování versus logování

Rozdíly mezi trasováním a logováním nelze vždy určit přesně, protože se mnohdy obě metody vzájemně prolínají. Hlavní rozdíly těchto dvou metod jsou následující:

### Logování

Většinou jej zkoumá administrátor. Logují se pouze důležité informace (např. neúspěšná instalace programu). Nesmí obsahovat mnoho opakujících se událostí, které nejsou potřebné. Většinou využívají standardní výstupní formulář a záznamy jsou většinou lokalizovány. Přidání nového typu události, stejně jako nové zprávy, nemusí být aktivní. Logování poskytuje informace užitečné pro diagnostiku a audit.

### Trasování

Trasováním se většinou zabývají pouze vývojáři. Logují se i méně podstatné informace (např. zamítnuté výjimky), a proto často obsahuje opakující se záznamy. Většinou je využit pouze omezený výstupní formulář. Lokalizace zapisovaných zpráv zpravidla chybí. Přidání nové zprávy nebývá aktivní. Trasování poskytuje informace užitečné k odstranění chyb v programu (užívány během vývojového cyklu i po uvolnění programu).

Proč není logování založené na kódování událostí vhodné pro trasování softwaru? Trasování je nízko úrovňové, to znamená, že je zde mnohem více typů zpráv, které musí být definovány, mnohé z nich jsou použity pouze na jednom místě ve zdrojovém kódu. Typy zpráv, které jsou logovány, jsou často méně stabilní. Výstup z trasování nemusí být lokalizován, protože se předpokládá, že jej bude používat vývojář. Je tedy důležité oddělovat trasovací zprávy od ostatních zdrojů, které vyžadují lokalizaci (jako zprávy o událostech). Jsou zde zprávy, které by neměly být nikdy zobrazovány. Přesto by trasovací zprávy měly být obsaženy ve zdrojovém kódu, protože mohou usnadňovat jeho čitelnost. Toto není vždy možné při logování událostí.

# 3 Log4net

Log4net je open source projekt, založený na velmi populární knihovně Log4j pro jazyk Java. Open source je počítačový program s přístupným zdrojovým kódem. Znamená to, že program je k dispozici v textové formě, ve které jej napsal programátor. A to obvykle každému. Díky tomu mají ostatní programátoři možnost studovat, jak program pracuje. Pomocí log4net je možné konfigurovat formát, typ logu, úroveň logu aj. s pomocí externího konfiguračního souboru a to i za běhu programu.

Log4net je framework, který usnadňuje programátorům zápis záznamů na různá místa - textový soubor, Event Log Windows, konzoli, databázi nebo na jiná umístění. V případě nějakého problému s aplikací je výhodné povolit logování a s jeho pomocí najít problém. Log4net umožňuje povolit logování za běhu aplikace, bez nutnosti jakkoli modifikovat zdrojový kód aplikace. Log4net je navržen tak, aby logovací příkazy mohly zůstat ve zdrojovém kódu bez snížení výkonu aplikace.

A co to vlastně framework je? Framework je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, návrhové vzory nebo doporučené postupy při vývoji.

## 3.1 Struktura log4net

Log4net využívá vrstvený přístup se čtyřmi základními vrstvami – *logger*, *appender*, *repository* a *layout*. V dalším textu budou popsány podrobněji.

### 3.1.1 Logger

*Logger* je komponenta zodpovědná za vlastní řízení logování jako celku. S touto komponentou komunikuje sledovaná aplikace a také zprávy, které jsou zaznamenávané. Její výstup je zobrazován pomocí komponenty *layout*, která bude popsána dále. Tento element poskytuje metody, pomocí nichž je možné zapsat jakýkoli záznam neboli log. Je možné vytvořit vícenásobný *logger* uvnitř naší aplikace. Každý l*ogger*, ke kterému je vytvořena instance, je udržovaný uvnitř log4net. To znamená, že není nutné posílat jeho instanci mezi různými třídami a objekty, pokud má být znovu použit. Namísto toho je možné jej volat pomocí jeho jména odkudkoli z aplikace. *Loggery* udržované uvnitř log4net rozlišují velká a malá písmena (jsou case-sensitive) a dodržují hierarchickou organizaci. Hierarchie je podobná jako při definování

jmenných prostorů v .NET[1]. Pro představu lze použít například situaci, kdy existují dva *loggery* definované jako a.b.c a druhý a.b. V tomto případě je a.b předchůdce pro a.b.c. Každý z nich má svůj rodičovský *logger,* ze kterého dědí. Na vrcholu této hierarchie se nachází výchozí, neboli kořenový l*ogger*, z něhož dědí všechny ostatní. Přestože takovéto pojmenovávání se doporučuje, je možné jej pojmenovat jakkoli. *Logger* je inicializován pomocí statické metody ze třídy log4net.LogManager.

### Úrovně logování zpráv

L*ogger* umožňuje určit jednu z pěti úrovní logování zpráv. Úroveň je instance třídy log4net.Core.Level. Následující přehled je setříděn podle priority od nejvyšší po nejnižší.

- OFF
- DEBUG
- INFO
- WARN
- ERROR
- FATAL
- ALL

Pokud *logger* nemá přiřazenu prioritu, automaticky ji dědí od svého rodiče. Pokud rodič také nemá nastavenou prioritu, postupuje se v hierarchii stále výš, až po kořenový *logger*. Kořenový *logger* má nastavenou výchozí hodnotu na DEBUG. Úroveň OFF znamená, že je logování úplně vypnuto. Úroveň ALL naproti tomu znamená, že se logují úplně všechny zprávy. Jsou to dvě krajní polohy, mezi nimiž je pět dalších úrovní. Při tom platí, že pokud je nastavena úroveň na WARN, budu logované hlášky varovné (WARN), chybové (ERROR) a kritických chyb (FATAL).

## 3.1.2 Appender

Každý „dobrý“ logovací framework musí umět zapisovat zprávy na různá místa. V log4netu se o to stará komponenta nazvaná *appender*. *Appender* umožňuje snadno zapisovat zprávy na různá místa - například do souboru, databáze, prohlížeče události atd. Pokud si nevystačíme s již předdefinovanými *appendery,* je možné si vytvořit svůj vlastní. Nově vytvořený *appender* musí implementovat rozhraní log4net.Appenders.IAppenderinterface. Jeden *logger* se nemusí omezovat pouze na jeden *appender,* ale může v něm být *appender* definovaný i vícekrát.

[1] .NET framework je rozsáhlá softwarová platforma, která je určena pro vývoj mnoha různých druhů aplikací. Za pomoci .NET frameworku je možné vyvíjet nejen klasické aplikace pro Windows, ale mimo jiné i webové aplikace a služby, aplikace pro mobilní zařízení a mnoho dalších. Více o .NET je možné najít třeba na stránkách Microsoft [15].

### 3.1.3 Filter

*Appender* obsahuje část nazvanou *filter*, která umožňuje určit, které zprávy budou skutečně zapsány, a které se zahodí. Ve jmenném prostoru log4net.Filter se nachází několik předdefinovaných filtrů. Jednotlivé filtry je možné skládat za sebe a vytvořit z nich řetězec na jednom appenderu. Ve výchozím nastavení jsou propuštěny filtrem všechny zprávy. Všechny filtry musí implementovat rozhraní log4net.Filter.IFilter.

### 3.1.4 Layout

Většinou nedostačuje pouhé nastavení umístění pro uložení našich zpráv, ale je potřeba také nastavit formát, v jakém má být práva uložena. To je zajištěno připojením *layout* k modulu *appender*. *Layout* je zodpovědný za formátování logovaných zpráv podle přání uživatele, zatímco *appender* se stará o zasílání již naformátovaných zpráv na požadované umístění. Pro vytvoření vlastního *layout* je nutné, aby tento *layout* dědil ze třídy log4net.Layout.LayoutSkeleton, která implementuje rozhraní ILayout. Jeden *appender* může mít přiřazen pouze jeden *layout*.

## 3.2 Log4net v praxi

Aby bylo možné začít log4net používat v aplikaci, je nutné nejprve nakonfigurovat jíž dříve zmíněné tři komponenty. Existují dvě možnosti, jak to udělat: buď je lze nakonfigurovat v odděleném souboru zvlášť od aplikace, nebo je možné konfigurovat je přímo v aplikaci. Lepší je použít oddělený soubor, protože tak je možné měnit konfiguraci logování, aniž by bylo nutné znovu kompilovat zdrojové soubory aplikace. Druhou, neméně podstatnou výhodou je, že toto nastavení je možné měnit přímo za běhu aplikace. Pokud bychom nechtěli vytvářet externí konfigurační soubor, je možné nastavení vložit přímo do aplikace, potom by stačilo vložit pouze tag *log4net*. Tag *configSection* je nezbytný pouze v případě, že je používán externí soubor. Následuje ukázka možné konfigurace konfiguračního souboru log4net, která je dále detailně rozebrána.

```
<?xml version="1.0" encoding="utf-8" ?>
  <configuration>
    <configSections>
      <section name="log4net"
      type="log4net.Config.Log4NetConfigurationSectionHandler,log4net-net-1.0" />
    </configSections>
    <log4net>
      <logger name="testApp.Logging">
        <level value="DEBUG"/>
      </logger>
      <root>
        <level value="WARN" />
        <appender-ref ref="LogFileAppender" />
        <appender-ref ref="ConsoleAppender" />
      </root>
      <appender name="LogFileAppender" type="log4net.Appender.FileAppender" >
        <param name="File" value="log-file.txt" />
        <param name="AppendToFile" value="true" />
        <layout type="log4net.Layout.PatternLayout">
          <param name="Header" value="[Header]\r\n"/>
          <param name="Footer" value="[Footer]\r\n"/>
          <param name="ConversionPattern"value="%d [%t] %-5p %c [%x]"  <%X{auth}> - %m%n" />
        </layout>
        <filter type="log4net.Filter.LevelRangeFilter">
          <param name="LevelMin" value="DEBUG" />
          <param name="LevelMax" value="ERROR" />
        </filter>
      </appender>
      <appender name="ConsoleAppender"type="log4net.Appender.ConsoleAppender" >
        <layout type="log4net.Layout.PatternLayout">
          <param name="ConversionPattern" value="%d [%t] %-5p %c [%x]" <%X{auth}> - %m%n" />
        </layout>
      </appender>
    </log4net>
  </configuration>
```

Ukázka kódu 1: Jednoduchý konfigurační soubor pro log4net.

### 3.2.1 Element logger

Element *logger* určuje individuální nastavení pro *logger*. Tento element může být definován pouze jako potomek elementu *log4net*.

```
<logger>
  <logger name="testApp.Logging"/>
  <level value="DEBUG"/>
  <appender-ref ref="LogFileAppender" />
  <appender-ref ref="ConsoleAppender" />
</logger>
```

Ukázka kódu 2: Definice elementu Logger.

| **Element logger** | |
|---|---|
| **Atribut** | **Popis** |
| name | Definuje jméno doggeru (povinný). |
| additivity | Může nabývat hodnot „true“, nebo „false“. Výchozí nastavení je na „true“, pokud nastavíme „false“, zakážeme loggeru dědit z jeho rodičovského loggeru. |

Tabulka 1: popis atributů elementu logger.

V elementu *logger* můžeme definovat následující elementy:

**Appender-ref** – tento atribut nemusí být definován vůbec, nebo jich může být definováno více; využívá jej *logger* k odkazování se na *appender* pomocí jeho jména.

**Level** – volitelný element. Definuje úroveň logování, *logger* potom akceptuje pouze událostí dané úrovně, nebo ty s vyšší prioritou.

**Param** – další parametry specifikující *logger*.

### 3.2.2 Element root

V konfiguračním souboru může být definován pouze jeden element *root* a musí to být potomek elementu *log4net*. V tomto nepovinném elementu je specifikován kořenový *logger*, ze kterého jsou všechny ostatní *loggery* odvozeny. Takže v případě, že nebudou v konfiguračním souboru definované jiné *loggery*, využije framework vlastnosti definované v tomto tagu.

```
<root>
  <level value="WARN" />
  <appender-ref ref="LogFileAppender" />
  <appender-ref ref="ConsoleAppender" />
</root>
```

Ukázka kódu 3: Definice elementu root.

Element *root* nepodporuje žádné elementy, pouze tři atributy, stejně jako *logger*: *appender-ref*, *level* a *param*, které již byly popsány.

### 3.2.3 Element appender

Každý *appender* musí mít jednoznačný název a musí mít specifikován typ, který implementuje. *Appender* může být definován pouze jako potomek elementu *log4net*.

```
<appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender" >
  <layout type="log4net.Layout.PatternLayout">
    <param name="ConversionPattern" value="%d [%t] %-5p %c [%x]"<%X{auth}> - %m%n" />
  </layout>
</appender>
```

Ukázka kódu 4: Definice elementu appender.

Element *appender* podporuje atributy *name* a *type*. Oby dva tyto atributy jsou povinné. Element *name* označuje jméno pro daný *appender*, které musí být jednoznačné v celém konfiguračním souboru. Jméno se používá k odkazování na *appender* pomocí elementu *appender-ref* jak již bylo uvedeno výše. Pomocí elementu *type* je specifikován typ pro nový *appender*. Mezi nejčastěji používané patři:

- log4net.Appender.FileAppender - zapisuje logy do souboru
- log4net.Appender.EventLogAppender - zapisuje logy do Event Logu Windows
- log4net.Appender.ConsoleAppender- zapisuje logy do konzole
- log4net.Appender.AdoNetAppender - zapisuje logy do databáze

*Appender* podporuje elementy *appender-ref*, *filter*, *layout* a *param*. Ty jsou rozebrány v dalším textu.

### 3.2.4 Element filter

*Filter* může být definován jedině v elementu *appender*. Podporuje pouze element *type*, který je povinný a určuje jméno typu filtru. Jediný parametr, který podporuje, je *param*.

Filtry mohou být uspořádány za sebe, čímž vytvoří řetězec. Každý z filtrů pak může danou událost přijmout, nebo zamítnout, nebo ji podstoupit následujícímu filtru. Pokud událost projde až na konec řetězce filtrů je takováto událost přijatá.

```
<filter type="log4net.Filter.LevelRangeFilter">
  <param name="LevelMin" value="DEBUG" />
  <param name="LevelMax" value="ERROR" />
</filter>
```

Ukázka kódu 5: Definice elementu filter.

Filtr definovaný v ukázce odmítne události, které mají úroveň nižší než Debug a vyšší než Error. To znamená, že budou zaznamenány pouze zprávy s úrovněmi mezi Debug a Error.

### 3.2.5 Element layout

*Layout* může být definován, obdobně jako *filter*, pouze jako potomek elementu *appender*. *Layout* podporuje jediný atribut *type*, který je povinný a definuje název typu pro tento *layout*. Může obsahovat jediný element *param*.

```
<layout type="log4net.Layout.PatternLayout">
    <param name="Header" value="[Header]\r\n"/>
    <param name="Footer" value="[Footer]\r\n"/>
    <param name="ConversionPattern"value="'%d [%t] %-5p %c [%x]"<%X{auth}> - %m%n" />
</layout>
```

Ukázka kódu 6: Definice elementu layout.

### 3.2.6 Element param

Element *param* může být součástí mnoha elementů, což je patrné z předchozího textu. Tento element podporuje tři atributy – *name*, *value* a *type*. Atribut *name* definuje jméno daného parametru a je povinný. Atributy *value* a *type* jsou nepovinné, ale vždy musí být definován nejméně jeden z nich.

```
<param name="ConversionPattern" value="%d [%t] %-5p %c [%x] "<%X{auth}> - %m%n />
```

Ukázka kódu 7: Definice elementu param.

### 3.2.7 Zápis logu

Jakmile je konfigurační soubor nastaven, je již velice snadné zapisovat logy přímo z dané aplikace. Následující ukázka zobrazuje, jakým způsobem je možné log z aplikace zapsat.

```
class Program {
    protected static readonly ILog log = LogManager.GetLogger(typeof(Program));
    static void Main(string[] args) {
        log4net.Config.XmlConfigurator.Configure();
        log.Warn("varovani");
        log.Debug("jede to");
        log.Error("chyba");
     }
}
```

Ukázka kódu 8: Zápis jednoduchého logu z aplikace

Dalších informace o této problematice je možné nalézt v literatuře [1] nebo [2].

# 4 Microsoft Enterprise Library

Enterprise Library je knihovna aplikačních bloků, které pomáhají vývojářům s často se opakujícími úlohami (jako jsou logování, validace, přístup k datům, obsluha výjimek a další). Enterprise library obsahuje i ukázky zdrojového kódu, který je možné libovolně modifikovat a používat ve vlastních projektech.

## Logging Application Block

Logging Application Block je jeden z aplikačních bloků Enterprise Library, který zjednodušuje implementaci nejčastějších funkcí používaných při logování. Vývojáři mohou využít Logging Block k zápisu informací na různá místa:

- Prohlížeč událostí
- Databáze
- Textový soubor
- Fronta správ
- Zprávy přes e-mail
- Zprávy Windows Management Instrumentation (WMI)
- Jiná umístění

Application Block poskytuje shodné rozhraní pro logování informací na různá místa. Díky tomu aplikační kód nemusí specifikovat umístění zapisované události. Konfigurační nastavení rozhoduje o tom, zda se daná informace zapíše, a také na jaké umístění a v jakém formátu se má informace zapsat. Tím je umožněno modifikovat chování logování, aniž by bylo nutné zasahovat do kódu aplikace.

## Kdy použít Logging Application Block

V případě, že aplikace vyžaduje zapisovat logované informace do Windows Event Log, do databáze, fronty zpráv, WMI, textového souboru nebo na e-mail, je dobré zvážit využití Logging Application Block k zajištění této funkcionality. Logging Application Block je výhodné použít také v případě, kdy je potřeba filtrovat logované zprávy podle jejich kategorie nebo priority, pokud je vyžadováno formátování zapisované zprávy, nebo je nutné měnit umístění, kam zapisujeme, beze změny kódu aplikace. Logging Application Block je navržený tak, aby byl rozšiřitelný, a také obsahuje nástroje umožňující tvorbu kódu.

## 4.1 Základní prvky Logging Application Block

Logging Application Block využívá čtyř základních bloků k zajištění logovacího frameworku: *logEntry*, *tracer*, *listener* a *formatter*.

### 4.1.1 LogEntry

*LogEntry* je objekt, který se „zapíše“, při logování zprávy. Kupříkladu, *LogEntry* v ukázce kódu 9 zapíše zprávu obsahující: Ahoj Světe. Takovýto kód implicitně vytváří objekt *logEntry* se zprávou Ahoj Světe. Metoda Logger.Write dovoluje specifikovat více informací, například:

**Category** – slouží k určení kategorie *logEntry*.

**Priority** – je využívána filtrem, pomocí priority můžeme nastavit, které logy mají být zapsány (výchozí hodnota je -1 což je minimální priorita).

**EventId** – hodnota pomocí které je možné logy rozdělit do kategorií (defaultně je nastaven na 0 pro uživatelem definované *logEntry* a na -1 pro *LogEntry* implicitně vytvořené pomocí metody Logger.Write).

**Severity** – specifikuje důležitost *logEntry*.

**Title** – název zprávy.

*LogEntry* obsahuje i další informace o stavu systému, čase, které jsou doplňovány automaticky (např. název počítače, čas, název aplikace, ID procesu atd.).

### 4.1.2 Tracer

Zatímco *logEntry* umožňuje logovat jednotlivé události, pomocí objektu *tracer* je možné zaznamenat, že zapisovaný log je součástí nějakého procesu. Například může být *logEntry* součástí procesu vytvoření nového zákazníka:

```
using (new Tracer("Vytvoreni noveho zakaznika")){
  Logger.Write("Ahoj Svete");
  }
```

Ukázka kódu 9: Vytvoření Tracer a zapsáni zprávy „Ahoj Světe“.

Jakmile je vytvořená instance *tracer*, může být doplněna o activityId (jako parametr konstruktoru), nebo povolit GUID a activityId bude vygenerováno automaticky. ActivityId je potom zapsáno do logu spolu s *logEntry*.

### 4.1.3 TraceListener

Když je *logEntry* „zapsán“, Logging Application Block jej nasměruje do jednoho, nebo více *traceListenes*. *TraceListener* je zodpovědný za uložení přijatého záznamu na odpovídající místo. Logging Application Block obsahuje devět předdefinovaných *traceListener*:

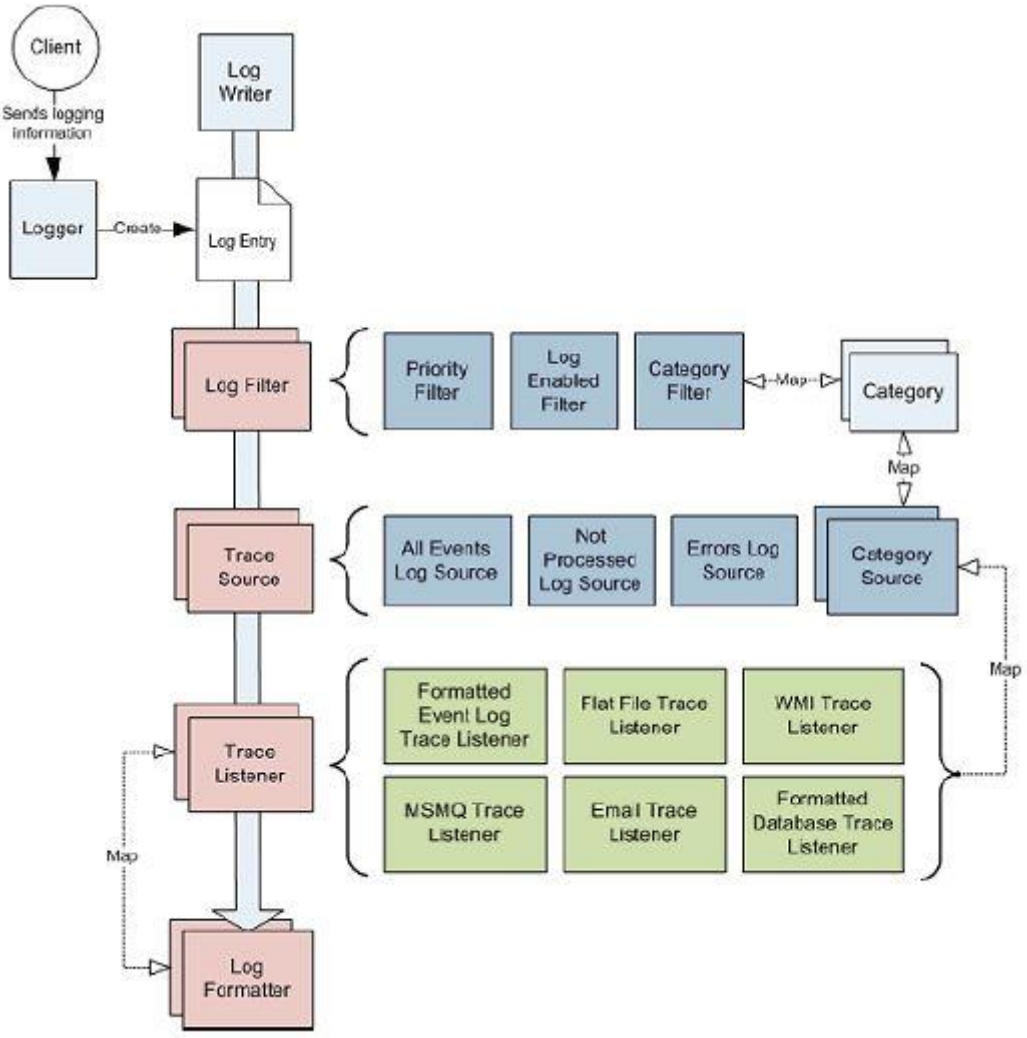
- Database TraceListener
- Email TraceListener
- FlatFile TraceListener
- Formatted EventLog TraceListener
- Msmq TraceListener
- Rolling Flat File TraceListener
- System.Diagnostics TraceListener
- WMI TraceListener
- XML TraceListener

Samozřejmě je možné vytvořit a nadefinovat si vlastní traceListener podle potřeb.

### 4.1.4 Formatter

*Formatter* slouží k formátování logu do tvaru, ve kterém je vyžadován na výstupu. Logging Application Block obsahuje dva typy formátovače. *TextFormatter*, který převádí *logEntry* na text. Druhým je *bingyLogFormatter*, který využívá *binaryFormatter* z frameworku .Net. Ten konvertuje *logEntry* do binární podoby.

## 4.2 Pořadí logovacího procesu




Obrázek 1: Schematický znázorněný postup jakým Logging Application Block zapisuje logy a objekty, které můžeme konfigurovat v jednotlivých fázích procesu.
Zdroj: msdn.microsoft.com, 2008.

Aplikace vytváří události, které Logging Application Block zapisuje pomocí objektu *logEntry*. *LogEntry* může definovat sadu kategorií, které se mapují do kategorií definovaných v konfiguraci. To je důležité pro posouzení, jak dále s *logEntry* nakládat, například při pozdějším průchodu přes filtry.

*LogWriter* propouští logy přes filtry, které byly definovány při konfiguraci. *LogFilter* může zablokovat logované záznamy na základě jejich priority, kategorie, nebo v případě, že logování není vůbec povoleno.

Pokud filtr logovaný záznam nezablokuje, vybere *logWriter* odpovídající t*raceSource*, na který logovaný záznam zašle. *TraceSource* je možné při konfiguraci nastavit mnoho vlastností. Existují také tři specialní *traceSource*, pomocí kterých je možné zajistit, aby byly zapsány opravdu všechny události (například pokud dojde k chybě při samotném logování, nebo *logEntry* nemá definovánu žádnou kategorii).

Následně může být definován jeden nebo více *traceListener* pro každý *traceSource*. Jinými slovy, je možné nakonfigurovat sadu *traceListener* pro každou kategorii, kterou může zpráva obsahovat. *Writer* se postará o to, aby se *logEntry* dostal k *traceListener*, který byl specifikován.

*TraceListener* následně využije *formatter* k převedení informace obsažené v *logEntry* do požadovaného formátu. Tím může být formátovaný text nebo binární kód a zapíše výsledek na výstup, který je specifikovaný typem *traceListener*. V závislosti na jeho typu může být výstup směrován do souboru, databáze, event logu, e-mailu atd.

## 4.3 Microsoft Enterprise Library v praxi

Konfigurace logování Enterprise Library se provádí nejčastěji pomocí konfiguračního souboru s příponou .config. Několik následujících stránek bude zaměřeno na popis jednotlivých elementů takového konfiguračního souboru.

Každý konfigurační soubor musí obsahovat následující sekci:

```
<configSections>
    <section name="loggingConfiguration" type="Microsoft.Practices.EnterpriseLibrary.Logging.
    Configuration.LoggingSettings, Microsoft.Practices.EnterpriseLibrary.Logging,
    Version=4.1.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
</configSections>
```

Ukázka kódu 10: Povinná hlavička konfiguračního souboru.

*ConfigSections* obsahuje jméno sekce loggingConfiguration a jméno třídy Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.LoggingSettings. Za ním následuje element *loggingConfiguration*, ve kterém jsou nakonfigurovány ostatní vlastnosti zápisu logů.

### 4.3.1 Element loggingConfiguration

Tento element specifikuje konfiguraci Logging Application Blocku. Element *loggingConfiguration* je povinný a v jeho těle jsou definovány všechny dále popsané elementy.

| **Element loggingConfiguration** | |
|---|---|
| **Atribut** | **Popis** |
| defaultCategory | Logy jsou zasílány do této kategorie, pokud není specifikována kategorie jiná (povinný). |
| tracingEnabled | Specifikuje, zda je povoleno trasování. Může nabývat hodnou true a false. Defaultně je nastaven na true. |
| logWarningsWhenNoCategoriesMatch | Specifikuje, zda výjimky mohou být zasílány do errors specialSource, pokud záznam obsahuje kategorii, která není specifikována v konfiguraci. Může nabývat hodnot true nebo false, defaultně je nastaven na true. |

Tabulka 2: Popis atributů elementu loggingConfiguration.

### 4.3.2 Element logFiltres

Element *logFilters* je potomkem *LoggingConfiguration* elementu. Jedná se o seznam filtrů, které zpracovávají logované zprávy ještě dříve, než jsou přesměrovány na trasovací zdroj. Element není povinný. Element *logFiltres* podporuje dva elementy - *add* a *categoryFilters*.

**Element add** slouží pro přidávání logovacích filtrů. Element je nepovinný, ale může se vyskytovat i vícekrát.

| Element add | |
|---|---|
| **Atribut** | **Popis** |
| name | Jméno logovacího filtru. Jméno musí být unikátní v celé sekci (povinný). |
| type | Název třídy, která implementuje rozhraní ILogFilter (povinný). |
| enabled | Specifikuje, zda je v dané aplikaci povolené logování. Tento atribut platí pro třídu LogEnabledFilter. Může nabývat hodnotu true, nebo false. |
| minimumPriority | Specifikuje nejnižší prioritu, kterou log musí mít, aby byl zapsán. Tento atribut platí pro třídu PriorityFilter. Nabývá celočíselných hodnot. Pokud je tento atribut vynechán, pak jsou logy zapisovány bez ohledu na minimální prioritu. |
| maximumPriority | Specifikuje nejvyšší prioritu, kterou log může mít, aby byl zapsán. Tento atribut platí pro třídu PriorityFilter. Nabývá celočíselných hodnot. Pokud je tento atribut vynechán, pak jsou logy zapisovány bez ohledu na maximální prioritu. |
| categoryFilterMode | Specifikuje, zda kategorie uvedené v categoryFilters jsou povoleny či nikoli. Tento atribut platí pro třídu CategoryFilter. Může nabývat hodnot AllowAllExceptDenied nebo DenyAllExceptAllowed (povinný). |

Tabulka 3: Popis atributů elementu add.

**CategoryFilters** je v podstatě seznam kategorií filtrů, které mohou být povoleny nebo zakázány. Je to volitelný element. Element *categoryFilters* podporuje jediný element add. Tento element slouží k přidávání nových kategorií.

### 4.3.3 Element categorySources

Element *categorySources* je potomkem l*oggingConfiguration* elementu. Slouží k definici trasovacích zdrojů, které směrují zaznamenávané položky do specifických kategorií *traceListeneru*. *categorySources* podporuje elementy add a *listener*.

**Element add** se používá k přidání nového trasovacího zdroje. Element je volitelný a může se vyskytovat vícekrát.

Element add podporuje atributy:

**name** – jméno kategorie logovaného záznamu, která přijímá daný trasovací zdroj. Jméno musí být jednoznačné v celé sekci a je povinně vyžadováno.

**switchValue** – natavuje vlastnosti kategorie *sourceLevel*. Specifikuje stupeň

závažnosti událostí, které jsou řízeny zdrojem trasování. Může nabývat hodnot

- ActivityTracing (povoluje Stop, Start, Suspend, Transfer, a Resume).
- All (povoleny jsou všechny výjimky).
- Critical (povoleny jsou jenom kritické události, tzn. fatal error, nebo pád aplikace).
- Error (povoluje kritické události a chyby – chyby jsou obnovitelné).
- Information (povoluje kritické události, chyby a informativní zprávy).
- Off (nepovoluje žádné události).
- Verbose (povoluje kritické události, chyby a varovné události, informativní události a události generované v průběhu překladu).
- Warning (povoluje kritické události, chyby a varovné události – nekritické problémy).

**autoFlush** – Specifikuje, zda má být událost zapsána automaticky hned po příchodu. V počátku je nastaven na true. Pokud je nastaven na false, je možné zapisovat položky, když budeme chtít.

**Element Listeners** blíže specifikuje *traceListerner*, do kterého jsou logy směrovány. Nejedná se o povinný element. Tento element podporuje element *add*, který slouží k přidávání nových *traceListener*.

### 4.3.4 Element specialSources

Specifikuje speciální trasovací zdroje, které mohou zpracovat přesné typy záznamu logovaných položek bez závislosti na použitých filtrech nebo jiných zdrojích. Tento element je povinný. Podporuje tři elementy – *errors*, *notProcessed* a *allEvents*.

**Element errors** popisuje chyby speciálního trasovacího zdroje. Tento

trasovací zdroj přijímá chyby a varování, které vznikly v průběhu logování. Element je povinný.

Podporuje atributy *name* a *switchValue* – jejich vlastnosti již byly popsány dříve a také jediný element *Listeners*, pomocí kterého může být rozšířena specifikace *Listener* definovaného jako potomka *loggingConfiguration*.

**Element notProcessed** - tímto trasovacím zdrojem jsou přijímány záznamy nacházející se v kategoriích, jež nebyly zpracovány jinými zdroji. Podporuje stejně jako element *errors* atributy *name* a *switchValue* a element *listeners*. Tento element není povinný.

**allEvent** - přijímá všechny záznamy, bez ohledu na to, zda již byly zpracovány jiným trasovacím zdrojem. Podporuje stejně jako elementy *errors* a *notProcessed* atributy *name*, *switchValue* a element *listeners*. Element není povinný.

### 4.3.5 Element listeners

Tento element obsahuje všechny *listeners*, které mohou být použity v dané aplikaci. Element je volitelný. Podporuje jediný element *add*. Pomocí tohoto elementu je možné přidat nové *listeners*. Je nepovinný, ale může se vyskytovat i vícekrát. Podporuje celou řadu atributů, mezi nejdůležitější z nich patří:

**name** – jméno nového *listener*, je povinné.

**type** – jméno typu třídy, která je odvozená od třídy *traceListeners*, atribut je povinný.

**message priorty** – určuje prioritu logu při přesunu do fronty a jeho pozici v cílové frontě. Může nabývat hodnot: Lowest, VeryLow, Low, Normal, AboveNormal, High, VeryHigh, Highest.

**filter** – vybírá zprávy předem nastavené úrovně z těch, které jsou detekovány.

Může nabývat hodnot:

- ActivityTracing (povoluje Stop, Start, Suspend, Transfer, a Resume).
- All (povoleny jsou všechny události).
- Critical (povoleny jsou jenom kritické události, tzn. fatal error nebo pád aplikace).
- Error (povoluje kritické události a chyby – chyby jsou obnovitelné).
- Information (povoluje kritické události, chyby a informativní zprávy).
- Off (nepovoluje žádné události).
- Verbose (povoluje kritické události, chyby a varovné události, informativní události a události generované v průběhu překladu).
- Warning (povoluje kritické události, chyby a varovné události – nekritické problémy).

**formatter** – název formátovače objektu *logEntry*, toto jméno musí být zahrnuto

v sekci *formatters*.

**fileName** – jméno souboru, který přijímá logované události.

### 4.3.6 Element formatters

Element *formatters* slouží ke specifikaci formátovače objektu *logEntry*. Je využíván formátovacím *traceListener*. Tento element je volitelný. Podporuje jediný element *add*, který přidává nové formátovače.

| **Element formatters** | |
|---|---|
| **Atribut** | **Popis** |
| name | Jméno formátovače, které musí být unikátní v celé sekci (povinný). |
| type | Typové jméno třídy, která implementuje rozhraní ILogFormatter (povinný). |
| template | Obsahuje šablony, včetně speciálních symbolů, které mohou být použity formátovačem. |

Tabulka 4: Popis atributů elementu formatters.

## 4.4 Zápis logu z aplikace

Po nastavení konfiguračního souboru lze již snadno zapisovat logy dané aplikace. Následující ukázka zobrazuje, jakým způsobem je možné log zapsat. V aplikaci můžeme samozřejmě specifikovat i některé parametry. Na následujícím příkladu je vidět mimo zápis vlastního logu také nastavení jeho priority a důležitosti.

```
class Program{
  static void Main(string[] args){
    LogEntry entry = new LogEntry(){
      Message = "Jede to :)",
      Priority = 10,
      Severity = System.Diagnostics.TraceEventType.Critical,
      Title = "muj log"
    };
    Logger.Write(entry);
  }
}
```

Ukázka kódu 11: Příklad zápisu logu z aplikace.

Na závěr je uvedena ukázka zdrojového kódu jednoduchého konfiguračního souboru. Je v něm nastaven jednoduchý filtr. Logy jsou zapisovány do textového souboru.

```
<configSections>
    <section name="loggingConfiguration"type="Microsoft.Practices.EnterpriseLibrary.Logging.
    Configuration LoggingSettings, Microsoft.Practices.EnterpriseLibrary.Logging, Version=4.1.0.0,
    Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
</configSections>
<loggingConfiguration name="Logging Application Block" tracingEnabled="true"
defaultCategory="General" logWarningsWhenNoCategoriesMatch="true">
     <listeners>
          <add source="Enterprise Library Logging" formatter="TextFormatter"log="Application"
          listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.
          FormattedEventLogTraceListenerData,Microsoft.Practices.EnterpriseLibrary.Logging,
          Version=4.1.0.0,Culture=neutral, PublicKeyToken=31bf3856ad364e35"
          traceOutputOptions="None" filter="All" type="Microsoft.Practices.EnterpriseLibrary.Logging.
          TraceListeners.FormattedEventLogTraceListener,Microsoft.Practices.EnterpriseLibrary.
          Logging, Version=4.1.0.0,Culture=neutral, PublicKeyToken=31bf3856ad364e35"
          name="Formatted EventLog TraceListener"/>
     </listeners>
     <formatters>
          <add template="Timestamp:{timestamp}&#xD;&#xA;Message:{message}&#xD;&#xA;
          Category: {category}&#xD;&#xA; Priority:{priority}&#xD;&#xA;
          EventId:{eventid}&#xD;&#xA;"type="Microsoft.Practices.EnterpriseLibrary.Logging.Formatte
          rs.TextFormatter, Microsoft.Practices.EnterpriseLibrary.Logging,Version=4.1.0.0,
          Culture=neutral, PublicKeyToken=31bf3856ad364e35" name="Text Formatter"/>
     </formatters>
     <logFilters>
          <add minimumPriority="1" maximumPriority="3" type="Microsoft.Practices.EnterpriseLibrary.
          Logging.Filters.PriorityFilter, Microsoft.Practices.EnterpriseLibrary.Logging, Version=4.1.0.0,
          Culture=neutral, PublicKeyToken=31bf3856ad364e35"name="Priority Filter" />
     </logFilters>
     <categorySources>
          <add switchValue="All" name="General">
               <listeners>
                    <add name="Formatted EventLog TraceListener" />
               </listeners>
          </add>
     </categorySources>
     <specialSources>
       <allEvents switchValue="All" name="All Events" />
       <notProcessed switchValue="All" name="UnprocessedCategory" />
       <errors switchValue="All" name="Logging Errors &amp;Warnings">
         <listeners>
           <add name="Formatted EventLog TraceListener" />
         </listeners>
       </errors>
     </specialSources>
</loggingConfiguration>
```

Ukázka kódu 12: Konfigurační soubor pro Logging Block.

Více o Enetrprise Library najdeme v publikacích [2], [3] a [4].

## 4.5 Log4net versus Enterprise Library

Oba dva frameworky jsou velmi oblíbené a tedy i úspěšné, který tedy zvolit?

Při výběru je nutné zvážit, které vlastnosti jsou pro konkrétní projekt důležité. Obecně lze mezi zásadní kritéria zařadit integraci s jinými programy a technologiemi, které jsou v projektu použity. V tomto ohledu je na tom o něco lépe Enterprise Library, která se skládá s několika bloků, mezi nimiž je integrace samozřejmostí, a v mnoha projektech tyto bloky také naleznou své místo. Enterprise Library rovněž přináší podporu WCF a jiných novějších technologií. Nové verze produktu Enterprise Library vychází mnohem častěji, čímž je zajištěna vyšší flexibilita a podpora nově vznikajícím technologiím.

Mezi hodně diskutovaný faktor patří výkon. Z tohoto hlediska je podle výsledků testů lepší produkt log4net. Log4net je v testech při zápisu několika set tisíc logů rychlejší o několik milisekund. V praxi je však tento rozdíl téměř zanedbatelný, protože tak velké množství záznamu v jednu chvíli není ve většině případu potřeba.

Co se týče stability nebo kvalitní dokumentace, jsou na tom obě technologie obdobě. Rozdíl je možné nalézt ještě při instalaci, která je log4net snazší, a také má mnohem menší nároky na volné místo na disku, což je dáno komplexnosti Enterprise Library.

# 5 System Center Essentials

System Center Essentials (SCE) je nástroj určený pro administrátory méně rozsáhlých sítí. Přináší jednotné prostředí pro vykonávání běžných denních úkolů, které v současné době správci IT buďto neprovádí, protože se jedná o časově náročné úkoly, anebo jsou těmito úkoly natolik vytíženi, že představují poměrně velkou část jejich pracovní náplně. SCE umožňuje pomocí jediné konzole sledovat „zdraví“ jednotlivých komponentů IT prostředí (od aktivních prvků přes servery až po složité IT služby), umožňuje provádět distribuce software na koncové stanice a v neposlední řadě provádět distribuci opravných balíčků. Přitom se neomezuje jen na produkty společnosti Microsoft.

Hlavními úkoly, jejichž řešení SCE usnadňuje, je proaktivní monitorování a diagnostika Windows serverů, klientů a aplikací, sledování a vyhodnocování stavu síťových zařízení. Dále je to správa aktualizací a jejich distribuce pro aplikace Microsoft i dalších výrobců a různé typy zařízení. Distribuci software je zajištěna v podobě MSI a EXE distribucí a to včetně aplikací třetích stran. V neposlední řadě je řešena inventarizace hardware a software s více jak třiceti zjišťovanými atributy, jako například dostupná velikost diskového prostoru, velikost RAM, instalované aplikace včetně jejich verzí. Zjišťování aplikací je omezeno na aplikace, které ukládají informace o instalaci do „Add/Remove programs“ („Přidat/Odebrat programy“) v ovládacích panelech.

## 5.1 Rodina produktů System Center

Microsoft System Center hraje klíčovou roli v dlouhodobé vizi společnosti Microsoft, jak IT organizacím pomocí samospravujících dynamických systémů usnadnit práci. Řešení System Center vyhledávají a shromažďují infoFrmace o infrastruktuře, zásadách, procesech a ověřených postupech, aby IT specialisté mohli optimalizovat IT struktury s cílem snížit náklady, zvýšit dostupnost aplikací a zlepšit poskytované služby.

Díky těmto integrovaným a automatizovaným řešením pro správu mohou být IT organizace produktivnější v poskytování služeb svým uživatelům. S cílem podpořit samospravovatelné dynamické systémy překlenují řešení System Center mezeru mezi vývojovými, provozními a IT odděleními – propojením lidí, procesů a nástrojů – prostřednictvím vyhodnocování jejich vzájemných závislostí a optimalizací výkonnosti obchodních procesů z hloubi operačních systémů, aplikací, služeb a pracovních postupů.

Do této rodiny spadá v současnosti celkem pět základních produktů:

- **System Center Configuration Manager** - umožňuje komplexně vyhodnocovat, nasazovat a aktualizovat servery, klientské počítače a zařízení ve fyzických, virtuálních, distribuovaných a mobilních prostředích.
- **System Center Operations Manager -** přináší ucelené řešení pro správu služeb. Představuje nejlepší volbu pro platformu Windows, protože bezchybně spolupracuje se softwarem a aplikacemi společnosti Microsoft, takže zvyšuje efektivitu při současné větší kontrole IT prostředí.
- **System Center Data Protection Manager -** představuje nový standard pro zálohování a obnovení systému Windows, zajišťující nepřetržitou ochranu dat pro aplikace a souborové servery pomocí integrace diskových a páskových médií. Díky pokročilé technologii nabízí produkt System Center Data Protection Manager podnikům všech velikostí rychlé a spolehlivé obnovení dat.
- **System Center Virtual Machine Manager** - představuje jednoduché a nákladově přijatelné řešení pro ucelenou správu fyzických a virtuálních počítačů, konsolidaci málo využívaných fyzických serverů a rychlé vytváření nových virtuálních počítačů využitím zkušeností a investic do serverových technologií společnosti Microsoft.
- **System Center Essentials** - je speciálně navržen pro středně velké firmy do 500 klientských počítačů a 50 serverů. System Center Essentials je uceleným řešením pro správu umožňující IT specialistům v organizacích střední velikosti proaktivně a efektivněji spravovat jejich IT prostředí.

Další informace o kterémkoli z těchto produktů lze najít na stránkách Microsoft TechNet [10]

## 5.2 Novinky v Systém Center Essentials 2010

Na podzim loňského roku byla uvolněna nová verze Essentials s označením System Cenetr Essentials 2010 ve verzi beta. Finální produkt by měl být k dispozici v průběhu roku 2010. V následující části textu bude nastíněno, jaká vylepšení nabízí nová verze na rozdíl od svého o tři roky staršího předchůdce.

Jedním z nejzajímavějších rozdílů je navýšení počtu serverů, které je možné spravovat. Zatímco ve verzi 2007 to bylo 30, nyní je možné spravovat až 50 serverů a 500 stanic a navíc až 100 jiných zařízení, jako například síťové prvky nebo tiskárny.

Nejvýraznější změny se však Essentials dočkal v oblasti virtualizace. V nové verzi Essentials je možné virtuální stroje vytvářet, kopírovat je z jiných serverů, provádět snímkování pro

případnou obnovu, anebo také inteligentně plánovat nové virtuální stroje. Pro snazší vytváření virtuálních strojů obsahuje Essentials řadu předem nakonfigurovaných šablon virtuálních strojů, které je možné využít.

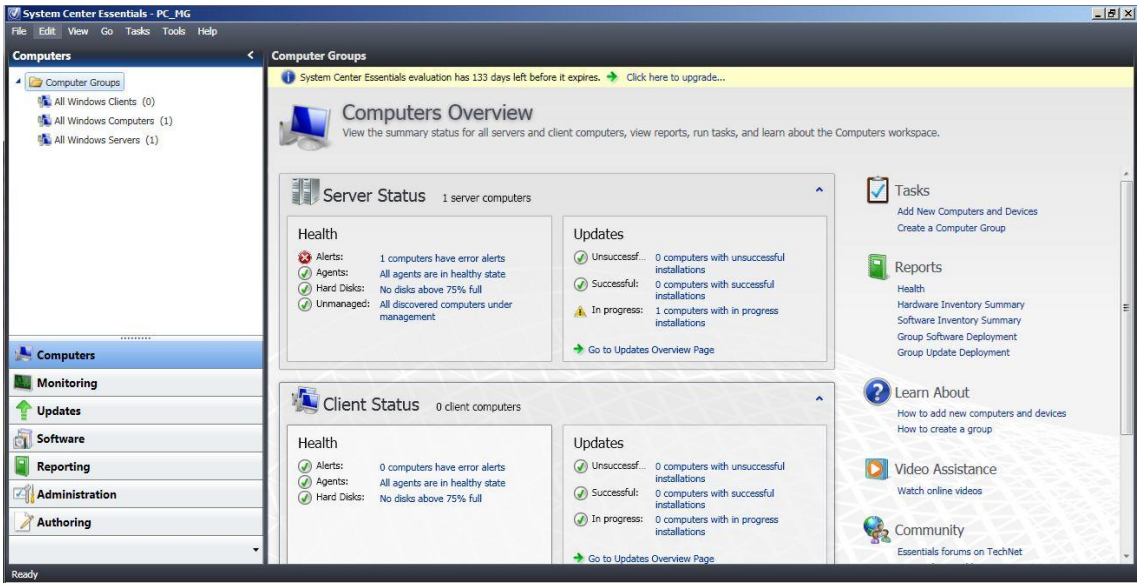
Co je nového v oblasti monitorování služeb? Essentials 2010 využívá pro svou znalostní bázi stejné management packy jako Operations Manager 2007. Management pack je soubor, který obsahuje definici prvků potřebných k monitorování dané aplikace, jako jsou health state, monitor rule nebo alert. Health state znázorňuje celkový stav systému. Monitor a rule slouží k získávání informací o systému a mohou vygenerovat alert, což je upozornění na změnu stavu systému.

Řídící balíčky je možné nově instalovat přímo z konzole, kde budeme také informováni, o nových řídících balíčcích. Další funkce, která usnadní každodenní práci s monitorováním prostředí, je možnost jednoduše vytvářet notifikační pravidla na základě událostí a výstrah v konzoli. Na vzniklé alerty je možné následně upozornit odesláním emailu, textové zprávy (SMS), nebo pomocí instant message (zaslání zprávy přes internet).

Novinky jsou také v oblasti nasazení nového software. S pomocí Essentials 2010 je možné distribuovat MSI nebo EXE balíčky, které se dají upravit podle parametrů. Celý proces je velice jednoduchý a je řízen průvodcem. Při distribuci je možné nově vybrat v průvodci filtr, tzv. Target System Type, v němž je možnost zvolit požadovanou architekturu, operační systém a jazyk.

Ve správě aktualizací přibyla možnost nastavení automatického stahování aktualizací pro vybrané produkty na základě prozkoumání prostředí. Samozřejmostí je i zachování původní manuální volby produktů, pro které je možno aktualizace stahovat. Nově je v Essentials funkce nazvaná „Auto-approve deadlines“, umožňující nastavit datum, do kterého budou automaticky nainstalovány všechny kritické události do klientských počítačů.

Cílem několika předchozích odstavců nebylo uvést všechny změny a novinky Essentials 2010 oproti jeho předchůdci. Jsou zde vyzdviženy jen některé nejvýznamnější inovace tohoto produktu. Kompletní seznam vylepšení je možné nalézt v technické dokumentaci k tomuto produktu [10].

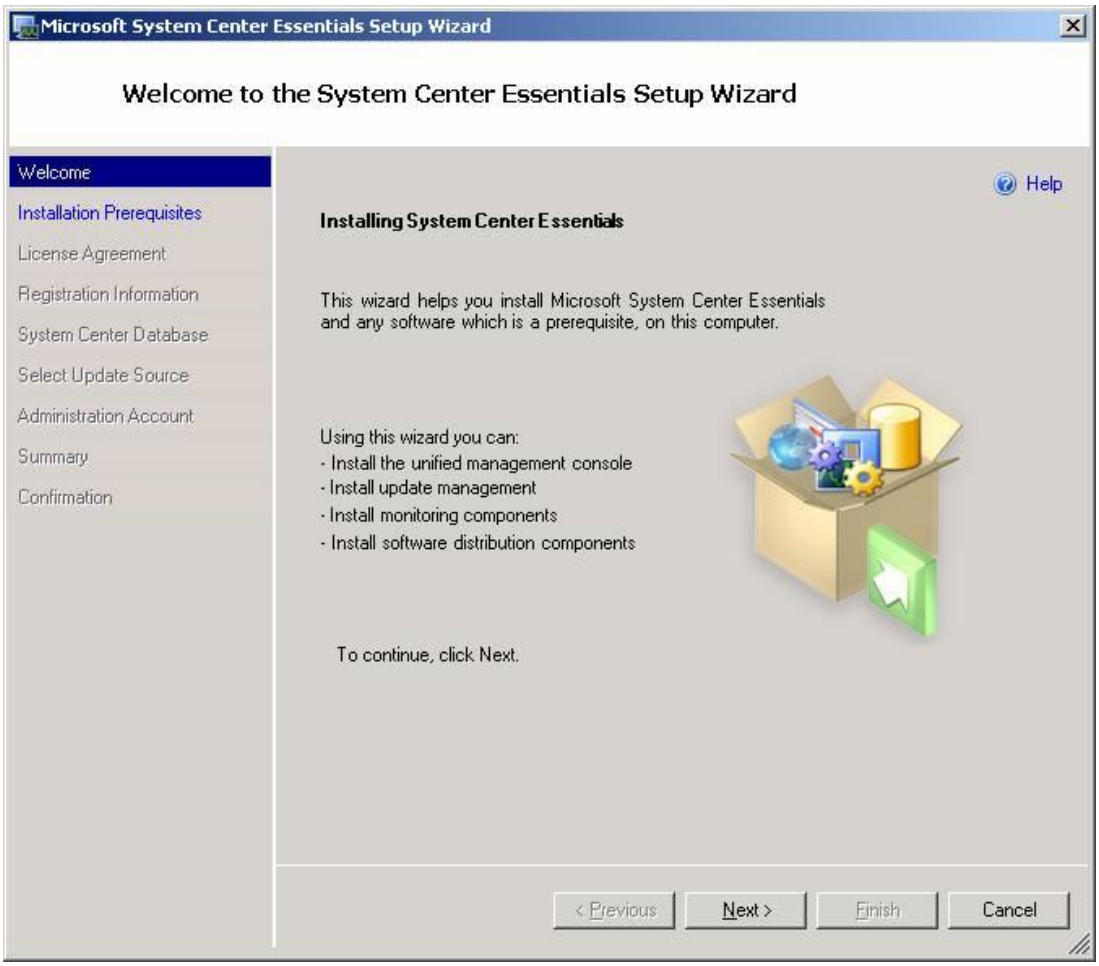



Obrázek 2: Náhled na prostředí Systém Center Essentials.

## 5.3 Instalace a konfigurace SCE 2010

Tato kapitola se bude zabývat základními kroky instalace a konfigurace SCE. Více informací o tomto tématu je možné získat například v dokumentu nazvaném Deployment guide, který je volně stažitelný na stránkách Microsoft [5].

Instalace produktu je snadná a umožňuje také instalaci na jediný server, kde se nachází databáze, instalační balíčky, ale volitelně zde mohou být i aktualizace stažené z internetu. K dispozici máme průvodce se systémovými požadavky, jenž nás trpělivě informuje o všech požadavcích, které je potřeba zajistit pro úspěšnou instalaci serveru.




Obrázek 3: Průvodcem řízená instalace SCE.

Doporučené hardwarové požadavky na server kde poběží SCE 2010 jsou následující:

- Procesor(y) s frekvencí 2,8 GHz nebo vyšší
- 4GB RAM
- 20 GB volného místa na disku (1 GB na systémovém disku); 150 GB volného místa na disku v případě, že budeme využívat virtualizaci

Minimální hardwarové požadavky na klienty spravované pomoci Essentials nepřevyšují požadavky operačního systému, na kterém běží.

Konfigurační proces Essentials je mnohem jednodušší než u jiných produktů této oblasti. Pro základní konfiguraci serveru jsou k dispozici průvodce. V prvním z nich je zahrnuta konfigurace požadovaných produktových vlastností, což obnáší definování nastavení proxy serveru, určení typu politik - lokálních nebo skupinových. Dále pak výjimky pro firewall a vestavěnou vlastnost remote assistance, která poskytuje vzdálené řízení pro počítače

s nainstalovaným operačním systémem Microsoft Windows XP a pozdější. Také je možno nakonfigurovat úložiště pro chybové reporty a možnost automatického zasílání chyb do společnosti Microsoft. Posledním krokem této fáze konfiguračního průvodce je nastavení reportování denních událostí o výstrahách, aktualizacích, software a inventárních datech.

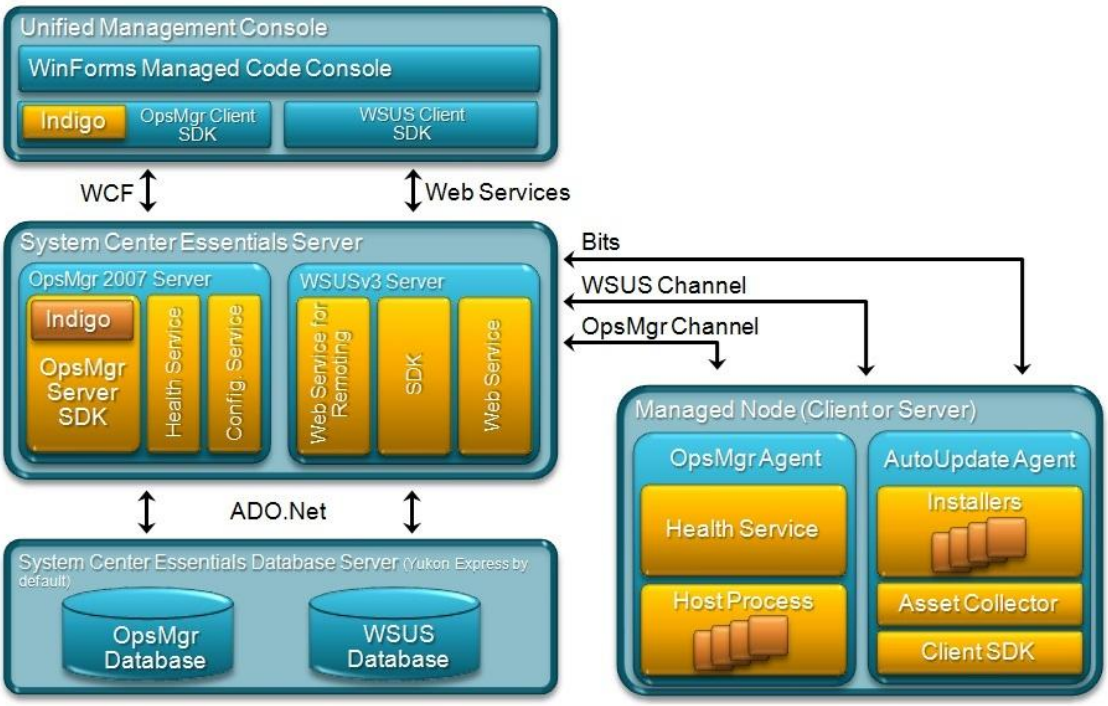
Více o System Center Essentials je možné nalézt v publikacích [10], [12], nebo [13].

## 5.4 Architektura System Center Essentials

Na následujícím obrázku si můžeme prohlédnout architekturu SCE. V klientské části se nachází Auto Update Agent, který zajišťuje distribuci aktualizací a jiných uživatelem vytvořených balíčku. Nalézá se zde také Health Agent, který v případě výpadku spojení se serverem sbírá potřebná data a následně po obnovení zabezpečeného spojení je odesílá na SCE server.

Na straně serveru je OpsMgr Server a dvě služby - Health Service a Config Service. Služba Health Service komunikuje se službou stejného názvu na straně klienta a zajišťuje zápis získaných dat do databáze. Databáze mohou být tří typů – operativní databáze, WSUS databáze a také reportovací databáze, která na schématu není zakreslena. Služba Config umožňuje konfiguraci chování agentů na klientské straně.

Napojení serveru na management konzoli je řešeno pomocí WCF, což umožňuje vytvořit na základě SDK konektor pro vlastní systém. Následná komunikace probíhá přes webovou službu.



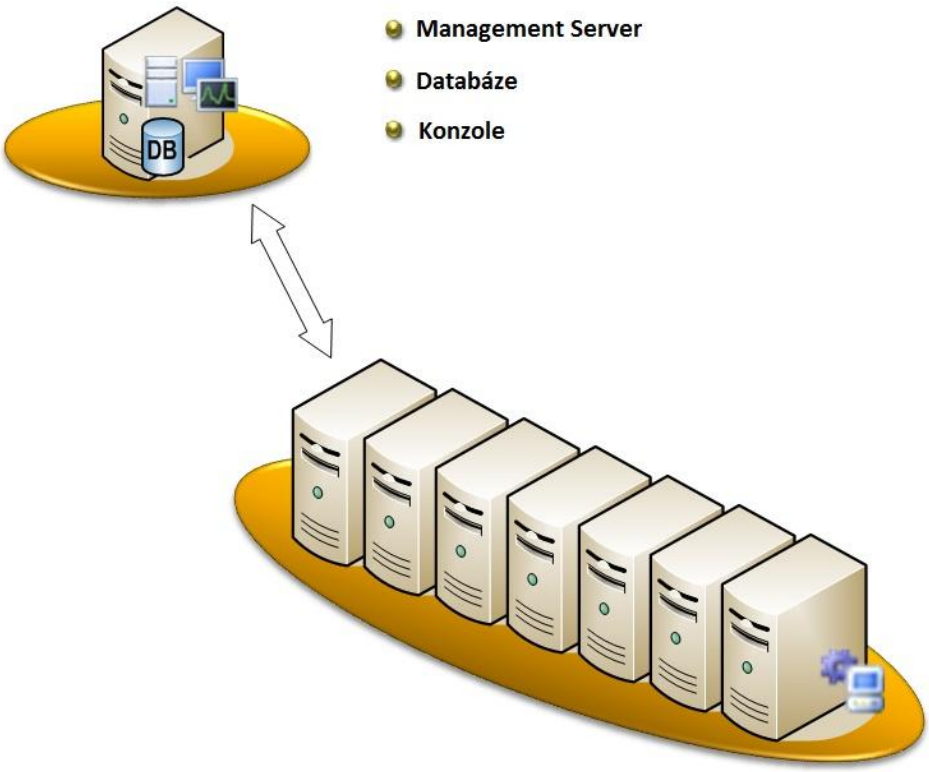

Obrázek 4: Architektura System Center Essentials 2007. Zdroj: David Mills, 2008.

## 5.5 Nasazení System Center Essentials

Systém Center Essentials je na rozdíl od System Center Operations Manager (SCOM) mnohem méně škálovatelný. System Center Operations manager je nástroj velice podobný SCE. Tento produkt je ovšem určen velkým firmám. Jde tedy o tzv. „dospělé řešení“ které obsahuje některé funkce a možnosti nad rámec SCE. Toto je ale samozřejmě vykoupeno jeho vyšší cenou. Management server SCE je možné instalovat jedině jako jeden celek, zatímco v SCOM je možné jej rozdělit na jednotlivé komponenty. Při nasazení SCE se většinou postupuje podle jednoho z následujících scénářů. Více je možné nalézt v literatuře [12], [13].

### 5.5.1 Single server topologie

V tomto scénáři máme nasazeny všechny komponenty (management server, konzoli a databázi) na jediném serveru.

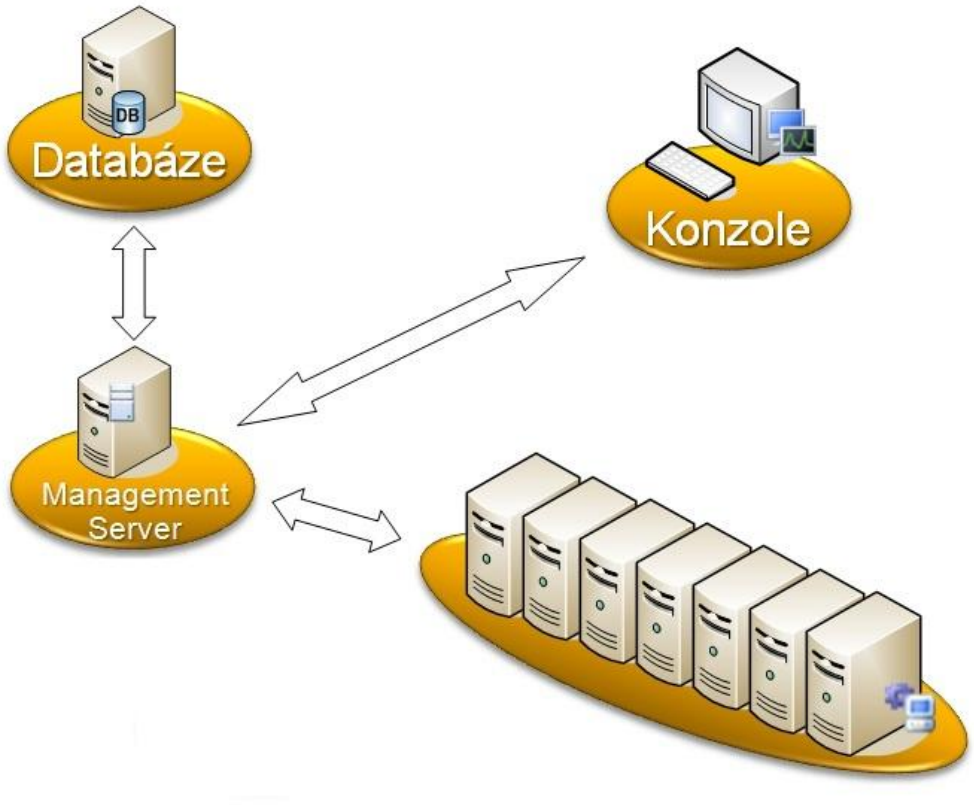



Obrázek 5: Schéma topologie Single Server. Zdroj: David Mills, 2008.

### 5.5.2 Distribuovaná topologie

V tomto případě může nastat situace, kdy se instaluje každá komponenta zvlášť, jak je znázorněno na obrázku níže. Druhá možnost je nainstalovat management server a konzoli na jeden server a databázi na jiný.

Tento způsob nasazení je výhodný, zejména pokud bude monitorováno více jak 200 počítačů. Díky oddělení jednotlivých komponent dosáhneme vyššího výkonu, než když jsou nainstalovány všechny na jednom serveru. Nesmí se však opomenout, že databázový server a management server musí být nainstalovaný ve stejné doméně.

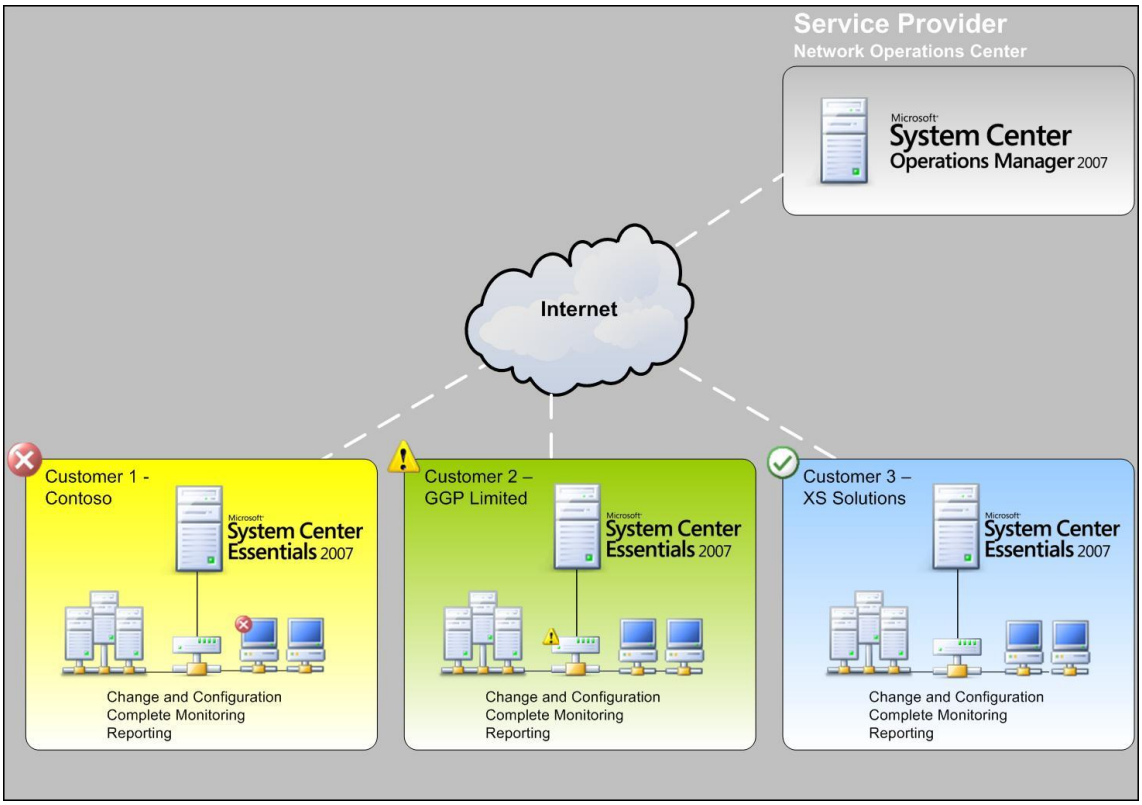



Obrázek 6: Schéma distribuované topologie. Zdroj: David Mills, 2008.

## 5.6 Vzdálená správa

Firmy, které se nezabývají jen prodejem svého softwaru nebo hardwaru, ale také jeho následnou podporou, jistě ocení možnost využít Systém Center Essentials ke vzdálené správě. K tomu je nutné využít druhý, již dříve zmíněný nástroj z rodiny produktů Systém Center nazvaný System Center Operations Manager. S pomocí kombinace těchto dvou produktů získáme velice silný nástroj, který umožní vzdáleně monitorovat, instalovat aktualizace nebo řešit jiné komplikace na straně klienta.

Více o této problematice je možné nalézt v literatuře [14].




Obrázek 7: Vzdálená správa. Zdroj: Vitalis Konopelec, 2008.

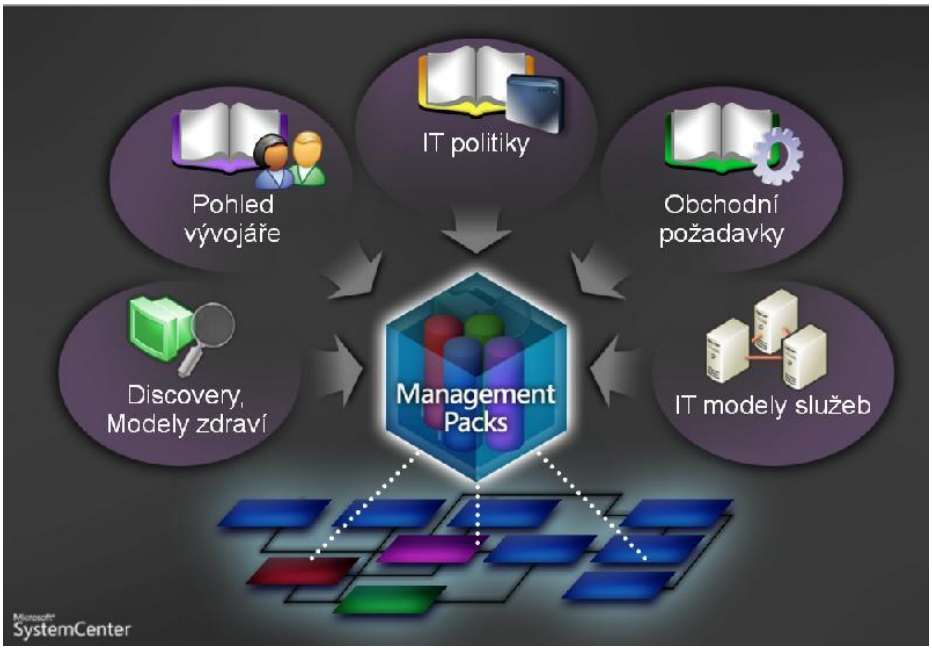
## 5.7 Management Pack

Management pack neboli řídící balíček představuje znalostní bázi pro monitorovanou aplikaci. Řídící balíček pro System Center Essentials je dokument psaný v eXtensible Markup Language (XML), který poskytuje podklady pro monitorování specifického software nebo hardware. Řídící balíčky obsahují definice různých komponent, ale také informace potřebné pro administrátora, aby mohl řídit danou aplikaci, zařízení nebo službu, které souhrnně označujeme objekt. Řídící balíčky fungují tak, že daný objekt nejprve naleznou a následně jej začnou monitorovat na základě modelu zdraví, který je také uložen v daném management packu.

V řídícím balíčku je popsáno, která data objektu zkoumat a provádět jejich analýzu, a na základě těchto dat SCE sestavuje obraz zdraví sledované komponenty.

Řídící balíček je možné si vytvořit, nebo si stáhnout a importovat již vytvořený řídící balíček ze stránek Microsoft - http://pinpoint.microsoft.com/en-US/default.aspx. Na těchto stránkách se

nachází katalog balíčků pro celou rodinu produktů System Center. Nalézají se zde balíčky například pro SQL, Windows server 2008, Windows 7 a mnoho dalších. Při vytváření vlastního řídícího balíčku je několik možností, jak na to. Jednou z nich je tvorba přímo v SCE s využitím Authoring konzole, kde je možné využít šablony a průvodce, což v začátcích tvorbu značně usnadní. Nevýhodou je, že tímto způsobem není možné vytvořit všechny objekty, které by řídící balíček mohl obsahovat. Jinou možností je vytvářet balíčky přímo psaním zdrojového XML. Poslední možností je využít některý z produktů třetích stran, zaměřeného na tvorbu řídících balíčků. Jedním z takových programů je například Silect MP Studio od společnosti Silect Software. Více informací o Silect Software lze nalézt v literatuře [7].




Obrázek 8: Prvky řídícího balíčku. Zdroj: Vitalis Konopelec, 2008.

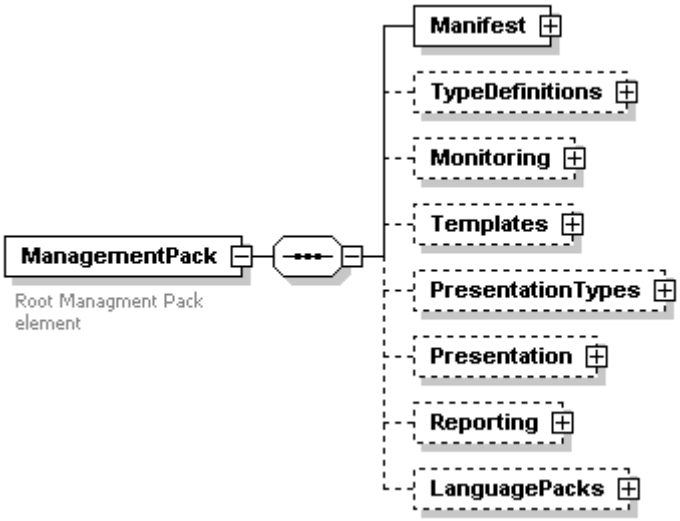
### 5.7.1 Výchozí řídící balíček

Je to jeden z řídících balíčků, které jsou importovány již při instalaci jako součást Essentials. Jde o speciální balíček, zprvu prázdný, ale jakmile je vytvořen nějaký objekt balíčku, například monitor, rule, nebo alert, je tento objekt uložen standardně právě do tohoto balíčku. Samozřejmě je možné daný objekt uložit do předem připraveného balíčku.

### 5.7.2 Struktura řídícího balíčku

Jako většina jazyků, i XML musí dodržovat předepsaný formát, aby jej mohl parser přečíst a následně byl vykonán. V našem případě musí řídící balíček dodržovat schéma pro tvorbu management packu, odpovídající běžnému XML dokumentu a také standardu W3C.

Schéma řídícího balíčku je rozděleno do osmi hlavních sekcí, jak je znázorněno na následujícím schématu. Schémata byla vytvořena pomocí nástroje Altova XMLSpy.




Obrázek 9: Náhled na schéma řídícího balíčku.

**Manifest** – manifest obsahuje identifikační údaje řídícího balíčku včetně jeho jména. Mimo to může obsahovat reference na jiné řídící balíčky. Tento element je povinný, všechny ostatní jsou volitelné.

**TypeDefinitions** – tento element se používá k popsání dostupných informací o monitorovaném objektu. Můžeme zde definovat další elementy - entity, data, schema a další.

**Monitoring** – v tomto elementu jsou definované pravidla, pomocí kterých je prováděno monitorování daného objektu. Mohou zde být definovány další atributy např. rules, tasks a jiné.

**Templates** – šablona poskytuje průvodce pro uživatele, jak nakonfigurovat monitoring pro daný cíl.

**PresentationTypes** – tento element definuje šablonu pro prezentaci získaných dat v SCE (např. State view, Event view, Diagram view).

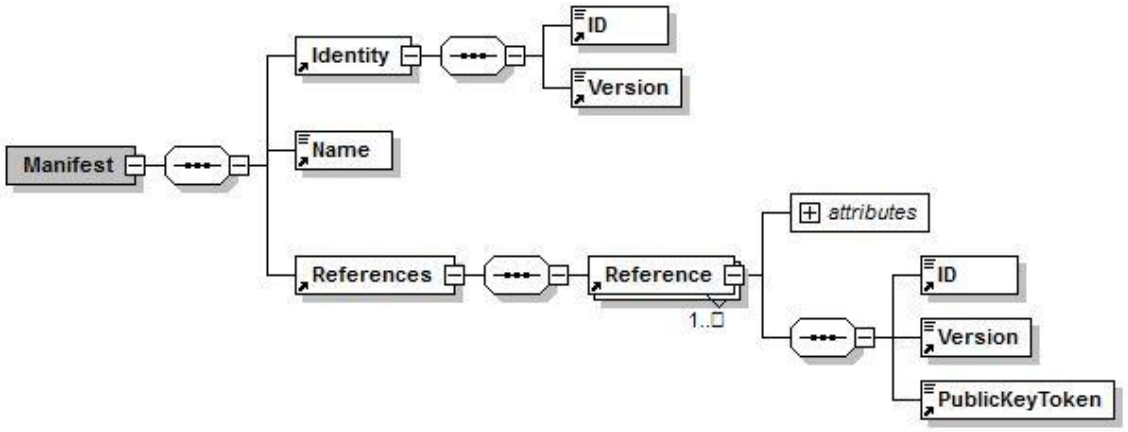
**Presentation** – definuje, která data se administrátorovi v konzoli zobrazí. Mohou to být například tasks, views, nebo jiné objekty.

**Reporting** – tento element slouží k definování reportů, které budou obsaženy v řídícím balíčku.

**Language Types** - umožňuje vytvořit balíček, jenž bude podporovat více jazyků.

### 5.7.3 Manifest

*Manifest* je první a zároveň jediný povinný atribut. Jeho účelem je především identifikace řídícího balíčku. Mimo to obsahuje reference na jiné balíčky, které je nutné importovat, aby daný balíček fungoval korektně. *Manifest* má tři elementy, jak je možno vidět na následujícím obrázku.




Obrázek 10: Schéma elementu manifest.

### Identity

Identita balíčku je tvořena jeho ID, verzí a případně klíčem, pokud je balíček podepsán. ID a verze jsou explicitně definovány v manifestu balíčku, zatímco veřejný klíč není znám do doby podepsání balíčku. Tento klíč může být závislý na certifikátu, pomocí kterého je balíček podepsán.

ID řídícího balíčku musí být stejné jako jeho název a musí dodržovat následující podmínky:

- může obsahovat znaky a až z (velké i malé), cifry 0-9, znak tečka „.“ a znak podtržítko „_“
- musí mít méně než 256 znaků
- musí být jedinečné napříč všemi elementy řídícího balíčku

Číslo verze řídícího balíčku je složeno ze čtyř numerických znaků. Každé číslo má délku mezi jedním a čtyřmi znaky. Žádné další omezení na zápis verzí není kladeno. Korektní zápis verze může tedy vypadat například:

- 1.0.0.0
- 1.0.15.0
- 12.1.1.1485
- 1234.1234.1234.1234

## Reference

Řídící balíčky mohou odkazovat na jiné již publikované balíčky. Pokud chceme vytvořit odkaz na jiný balíček, musíme specifikovat:

- ID řídícího balíčku
- Minimální verzi balíčku, kterou vyžadujeme
- Veřejný klíč (pouze v případě že je balíček podepsán)

Jakmile máme řídící balíček specifikovaný, vytvoříme pro něj alias. Kdykoli budeme chtít odkázat na některý z jeho elementů, použijeme jako prefix jeho alias a znak „!“.

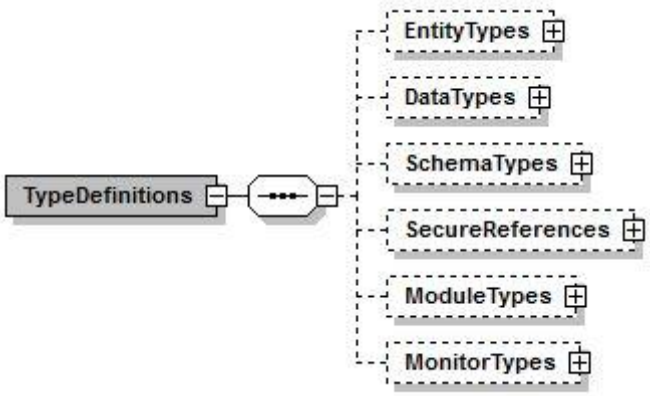
Pro každý balíček je nutné použít unikátní alias a zároveň jich nesmíme vytvořit více pro jeden balíček.

```
<Manifest>
  <Identity>
   <ID>Sample.MyFirstMP</ID>
   <Version>1.0.0.0</Version>
  </Identity>
  <Name>My First MP</Name>
  <References>
   <Reference Alias="Windows">
    <ID>Microsoft.Windows.Library</ID>
    <Version>6.2.4589.0</Version>
    <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
   </Reference>
   <Reference Alias="SC">
    <ID>Microsoft.SystemCenter.Library</ID>
    <Version>7.1.3355.2</Version>
    <PublicKeyToken>31bf3856ad364e35</PublicKeyToken>
   </Reference>
  </References>
</Manifest>
```

Ukázka kódu 13: Definice elementu manifest.

### 5.7.4 TypeDefinitions

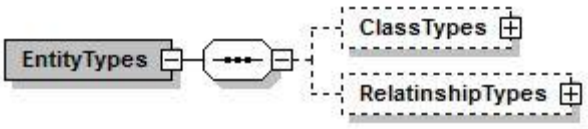
*TypeDefinitions* slouží k popisu monitorovaného objektu a skládá se ze šesti elementů, které budou v následujícím textu rozebrány podrobněji.




Obrázek 11: Schéma TypeDefinitions.

## EntityTypes

Tato sekce slouží k definici typů objektů, které mají být objeveny a případně i monitorovány, ale také vztahů mezi nimi. V této sekci je možné definovat strukturu aplikace, kterou chceme monitorovat.




Tabulka 5: Schéma elementu EntityTypes.

### Element ClassTypes

Slouží k definování typu objektů, které mají být nalezeny a následně mohou být monitorovány.

Obsahuje atributy:

| **Element ClassTypes** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |
| Abstract | Definuje, zda je třída abstraktní nebo konkrétní. Abstraktní třídy nemohou být instalovány (povinný). |
| Base | Definuje, ze které třídy tato třída dědí (povinný). |
| Hosted | Definuje, zda instance této třídy jsou hostovány jinou třídou. |
| Singleton | Definuje, jestli je tato třída unikátní. |

Tabulka 6: Popis atributů elementu ClassTypes.

### Element RelationshipType

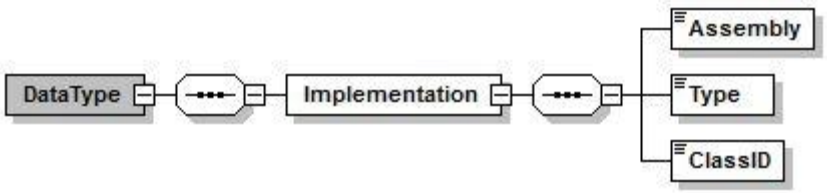
Element *RelationshipType* slouží k definici vztahů mezi typy tříd.

| **Element RelationshipType** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |
| Abstract | Definuje, zda je třída abstraktní nebo konkrétní. Abstraktní třídy nemohou být instalovány (povinný). |
| Base | Definuje, ze které třídy tato třída dědí (povinný). |

Tabulka 7: Popis atributů elementu RelationshipType.

## Data Types

Tento element definuje typ dat, která si mohou moduly Essentials mezi sebou předávat.




Obrázek 12: Schéma elementu DataType.

Obsahuje atributy:

| **Element DataTypes** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |
| Base | Definuje, ze které třídy tato třída dědí (povinný). |

Tabulka 8: Atributy elementu DataTypes.

```
<DataType ID="System.Performance.SignatureEntry" Base="System!System.BaseData" Accessibility="
Public">
  <Implementation>
    <ClassID>RFG5741B-7DE5-7C66-948F-651554968E0F8</ClassID>
  </Implementation>
</DataType>
```

Ukázka kódu 14: Definice elementu DataType.

## SchemaType

Využívá se k vytvoření znovupoužitelného schématu, které je možné použít pro definování typu monitoru nebo modulu. Dá se využít, pokud více různých typu monitoru, nebo modulu používá opakovaně stejnou část definice.


Obrázek 13: Schéma elementu SchemaType.

| **Element SchemaType** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |

Tabulka 9: Popis atributů elementu SchemaType.

```
<SchemaType ID="Microsoft.Windows.RegistryAttributeDefinitionsSchema" Accessibility="Public">
  <xsd:complexType name="RegistryAttributeDefinitionsType">
    <xsd:sequence>
          <xsd:element name="RegistryAttributeDefinition" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="AttributeName" type="xsd:ID"/>
          <xsd:element name="Path" type="xsd:string"/>
          <xsd:element name="PathType" type="xsd:integer"/>
          <xsd:element name="AttributeType" type="xsd:integer"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</SchemaType>
```

Ukázka kódu 15: Definice elementu SchemaType.

### SecureReference

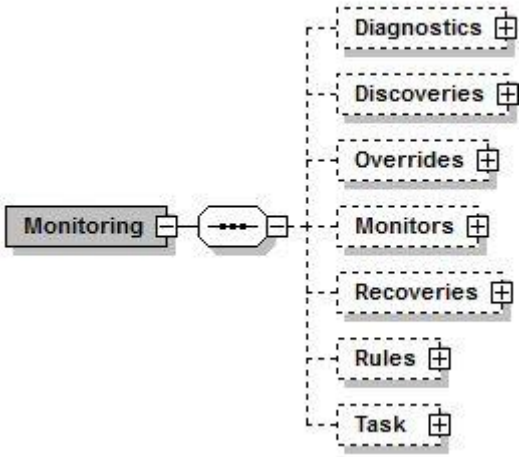
V rámci řídícího balíčku je možné nastavit *SecureReference*. *SecureReference* umožňuje zvýšit práva účtu agenta a díky tomu spouštět procesy, na které bez korektního nastavení *SecureReference* nemá agent dostatečná práva. Pro správnou funkci *SecureReference* je nutné, aby uživatel po importování řídícího balíčku vytvořil asociaci mezi účtem a příslušnou *SecureReference*.

| **Element SecureReference** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |

Tabulka 10: Popis atributů elementu SecureReference.

## 5.7.5 Monitoring

Tato sekce slouží k popisu, jak daný objekt sledovat a jakým způsobem vyhodnocovat získaná data. Rozpadá se na sedm základních elementů, jak je znázorněno na následujícím schématu.

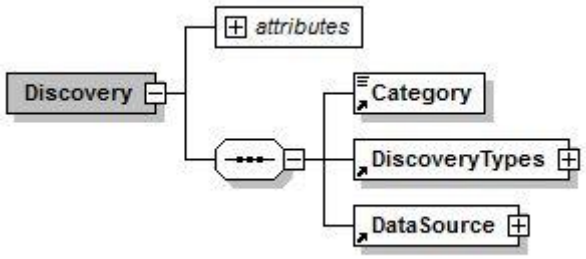



Obrázek 14: Schéma sekce monitoring.

### Discovery

Objekt discovery se v SCE využívá k vyhledání objektů, které mají být monitorovány v síti. Protože vývojáři řídících balíčků nemohou vědět, jaké síťové prvky se nacházejí v síti, kde bude řídící balíček nasazen, je definován pouze typ objektu, který má řídící balíček monitorovat.

Objekt discovery může k hledání síťových prvků využívat registry, wmi, skripty, OLE DB, LDAP, nebo dokonce vlastní kód. Pokud objekt discovery nalezne na síti objekt, který nemá být monitorován, máme možnost omezit jeho rozsah pomocí *Overrides*.




Obrázek 15: Schéma elementu Discovery.

| **Element Discovery** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Cílový objekt (povinný). |
| ConfirmDelivery | Definuje, zda modul má upozornit zdroj dat, že jeho data přijal. |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| Priority | Element pouze pro vnitřní použití. |

Tabulka 11: Popis atributů elementu Discovery.

### Element Category

Určuje kategorii tohoto modulu. Hodnota tohoto atributu je vždy nastavena na Discovery.

### Element DiscoveryTypes

Tento element definuje třídy a vazby, které mají být rozpoznány. Definuje dva Elementy – *DiscoveryClass* a *DiscoveryRelationship*. Tyto elementy mají jediný atribut *TypeID* a také pouze jeden element *Property*. *Property* může definovat dva atributy *TypeID* a *PropertyID*.

```
<DiscoveryTypes>
  <DiscoveryClass TypeID="Demo.Application">
    <Property PropertyID="Version"/>
    <Property PropertyID="Path"/>
    <Property TypeID="System!System.Entity" PropertyID="DisplayName"/>
  </DiscoveryClass>
  <DiscoveryRelationship TypeID="Windows!Microsoft.Windows.ComputerHostsLocalApplication"/>
</DiscoveryTypes>
```

Ukázka kódu 16: Definice elementu DiscoveryTypes.

### Element DataSource

Tento element slouží k definici zdroje dat. Může obsahovat tři elementy.

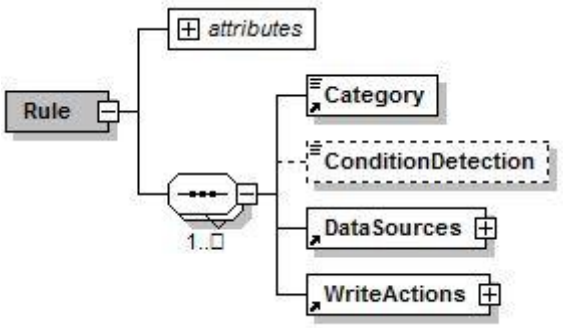
| **Element DataSource** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| TypeID | ID datového zdroje, který je používán. |

Tabulka 12: Popis atributů elementu DataSource.

## Rules

V SCE se rules neboli pravidla používají ke sběru dat. Daty můžeme rozumět například události, které generují sledované objekty. Pravidla mohou být využívány i ke generování výstrah (alert), pokud se změní zdravotní stav monitorovaného objektu. Touto vlastností se podobají monitorům.

Pravidla se dají využít například ke sběru konkrétních událostí z aplikačního logu počítačů se systémem Windows. Tato sbírka událostí je uložena v databázi SCE, kde může být analyzována pomocí reportů a pohledů.




Obrázek 16: Schéma elementu Rule.

| **Element Rule** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Cílový objekt (povinný). |
| ConfirmDelivery | Definuje, zda modul může upozornit zdroj dat, že data přijal. |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| DiscardLevel | Pouze pro vnitřní použití. |

Tabulka 13: Popis atributů elementu Rule

### Element Category

Tento element musí nabývat některou z následujících hodnot:

- PerformanceCollection
- Operations
- EventCollection
- StateCollection
- SoftwareAndUpdates
- Alert
- Custom
- AvailabilityHealth
- PerformanceHealth
- ConfiguraitonHealth
- SecurityHealth
- Notification
- Maintenance

### Element DataSources

Element *DataSources* definuje datový zdroj. Datových zdrojů může být definováno více, v tom případě ale musí být definován atribut *ConditionDetection*, aby byl zajištěn správný počet vstupů.

| **Element DataSources** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| TypeID | ID datového zdroje, který je používán. |

Tabulka 14: Popis atributů elementu DataSources.

### Element ConditionDetection

Musí být definován pouze v případě, že je navázáno více datových zdrojů, a musí akceptovat korektní počet datových vstupů. Tento element má shodné atributy jako *DataSources.*

### Element WriteActions

V rules musí být definován nejméně jeden tento atribut. Tento modul může sloužit například k zápisu dat do databáze, spuštění skriptu, nebo vygenerování výstrahy.

| **Element WriteActions** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| RunAs | Definuje zabezpečené reference. |
| Target | Cílový objekt (povinný). |
| TypeID | ID modulu akce, ze kterého tento modul dědí (povinný). |

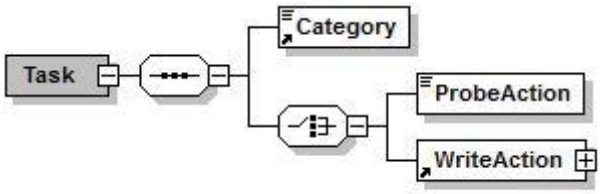
Tabulka 15: Popis atributů elementu WriteActions.

## Tasks

Systém Center Essentials 2010 umožňuje spouštět předdefinované úlohy, které jsou importované s řídícími balíčky, nebo které si administrátor sám vytvoří. Pokud má být úloha vytvořena, je možné zvolit agent task – v tom případě může být úloha spouštěna vzdáleně a to jak na spravovaném serveru, tak na počítači s nainstalovaným klientem. Druhou možností je console task, a v tom případě může být úloha spuštěna pouze na lokálním počítači. V Essentials 2010 může být dávkový soubor nebo skript spouštěn podobně jako úloha vzdáleně i lokálně, ale pokud je úloha vygenerována výstrahou nebo událostí, pak je možné ji spustit pouze lokálně.

Při vytváření úloh jsou k dispozici následující typy:

- Command line – umožňuje spustit dávkový soubor nebo aplikaci na agentu, nebo serveru. Tento typ úlohy může běžet lokálně i vzdáleně.
- Run a skript – umožňuje spustit skript na agentu, nebo serveru.
- Allert command line – umožňuje spustit úlohu, pokud je vygenerováno specifikované varování.
- Event command line – umožňuje spustit úlohu, pokud je vygenerována specifikovaná událost.




Obrázek 17: Schéma elementu Task.

| **Element Task** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Cílový objekt (povinný). |
| Timeout | Definuje čas, po který může task běžet. |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |

Tabulka 16: Popis atributů elementu Task.

**Elementy Category a WriteAction** jsou popsány v sekci *Rules*.

### Element ProbeAction

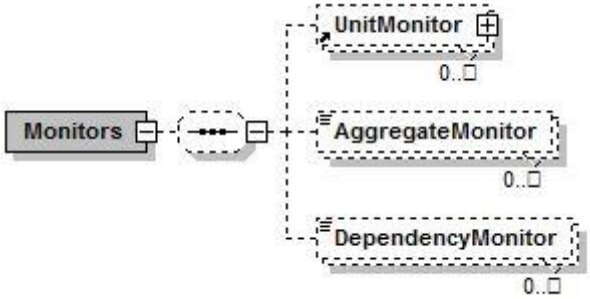
Tento element se používá, pokud jde o úlohu zaměřenou na diagnostiku, tzn., nemění se stav systému.

| **Element ProbeAction** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| TypeID | ID modulu, ze kterého ProbeAction dědí (povinný). |

Tabulka 17: Popis atributů elementu ProbeActions.

## Monitors

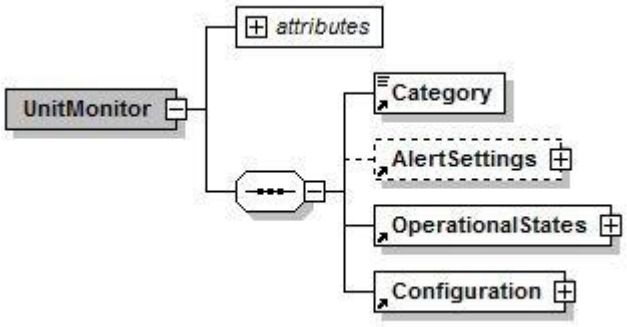
Monitory se v SCE využívají k posuzování rozmanitých situací, které mohou nastat v monitorovaném objektu. Je možné je využít, pokud chceme například posuzovat hodnoty výkonnostního čítače, sledovat existenci události, výskyt nových dat v logu nebo výskyt Simple Network Management Protocol (SNMP) pastí. Výsledek těchto hodnocení určuje zdravotní stav sledovaného cíle a upozornění, které jsou generovány. K dispozici máme tři typy monitorů: *UnitMonitor*, *AggregateRollupMonitor*, a *DependencyRollupMonitor*.




Obrázek 18: Schéma elementu Monitors.

### UnitMonitor

*UnitMonitor* je základní monitorovací komponenta. Využívá se k definování stavů monitorovaných objektů a popisu, jak tyto stavy detekovat. Jedním z nejjednodušších je například Basic Service Monitor, který je schopen sledovat, zda je spuštěná konkretní služba ve Windows. Komplexnější monitory mohou spouštět skripty, sledovat výkonnostní čítače, prověřovat logovací soubory. *UnitMonitor* může být přidán k *AggregateRollup* nebo *DependencyRollupMonitorům*. Pomocí tohoto monitoru můžeme také generovat upozornění.




Obrázek 19: Schéma elementu UnitMonitor.

Element UnitMonitor může definovat atributy:

| **Element UnitMonitor** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |
| Enabled | Definuje, zda ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Určuje cílový objekt. (povinný). |
| ParentMonitorID | Monitor, kterému předává svůj zdravotní stav (povinný). |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| Priority | Priorita, většinou nastavena na „normal“. |
| RunAs | Zabezpečené reference. |
| TypeID | Odkazuje na definici typu monitoru. |

Tabulka 18: Popis atributů elementu UnitMonitor.

### Element Category

Jako kategorii volíme jeden z vysokoúrovňových monitorů (Availability, Configuration, Performance, nebo Security).

### Element OperationalStates

Tato sekce slouží k definování stavů, ve kterých se monitor může nacházet. Například pokud je monitorována služba, která může být spuštěná nebo zastavená, lze jí nadefinovat dva stavy monitoru nazvané služba běží a služba stojí.

| **Element OperationalStates** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| MonitorTypeStateID | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| HealthState | Zdravotní stav monitoru, může nabývat hodnot Error, Warning, Success (povinný). |

Tabulka 19: Popis atributů elementu OperationalState.

```
<OperationalStates>
   <OperationalStateID= "ComponentProblem"HealthState= "Warning"MonitorTypeStateID=
   "FirstEventRaised"/>
   <OperationalStateID= "ComponentOK"HealthState= "Success"MonitorTypeStateID=
   "SecondEventRaised"/>
</OperationalStates>
```

Ukázka kódu 17: Ukázka nastavení stavu OperationalState.

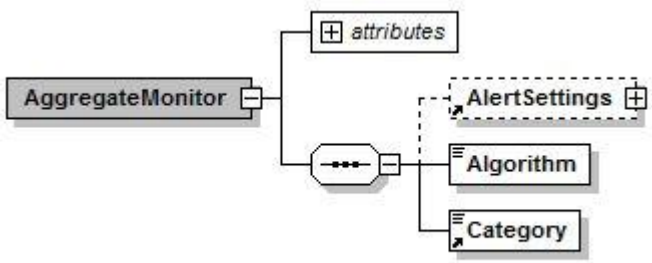
### AggregateRollupMonitor

Tento typ monitoru odráží stav *UnitMonitor*, *DependencyRollupMonitor* nebo jiného *AggregateRollupMonitor* zacíleného na objekt. Tento monitor se obvykle využívá k seskupení více různých monitorů pod jeden a tento monitor je následně využíván k nastavení zdravotního stavu a generování upozornění podle jednoho z dvojice algoritmů:

- Worst of
- Best of.

Všechny cíle, které je možné v Essentials 2010 monitorovat, spadají pod jeden z následujících vysokoúrovňových *AggregateRollupMonitor*:

- Availability
- Configuration
- Performance
- Security




Obrázek 20: Schéma elementu AggregateMonitor.

| Element AggregateMonitor | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Cílový objekt. (povinný). |
| ParentMonitorID | Monitor, kterému předává svůj zdravotní stav (povinný). |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| Priority | Většinou nastaven na normal. |
| RunAs | Zabezpečené reference. |

Tabulka 20: Popis atributů elementu AggregateMonitor.

Element category

Jako kategorii volíme jeden z vysokoúrovňových monitorů (Availability, Configuration, Performance, nebo Security).

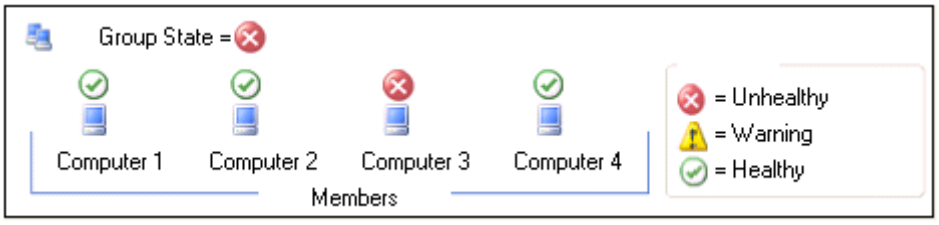
Element algorithm

Tento element nemá žádné další atributy, obsahuje přímo jméno daného algoritmu. Tento algoritmus rozhoduje, podle jakých pravidel bude nastavován healts state.

Rozlišujeme dva typy algoritmů:

1. WorstOf:

Tento algoritmus zobrazí stav monitoru podle objektu s nejhorším stavem.



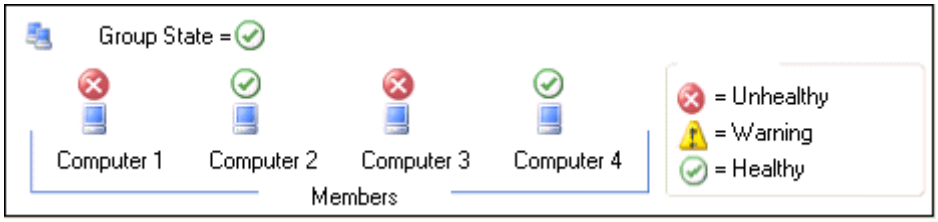

Obrázek 21: Algoritmus "WorstOf".

```
<Algorithm>WorstOf</Algorithm>
```

Ukázka kódu 18: Nastavení WorstOf algoritmu.

2. BestOf:

Tento algoritmus zobrazí stav monitoru podle objektu s nejlepším stavem.



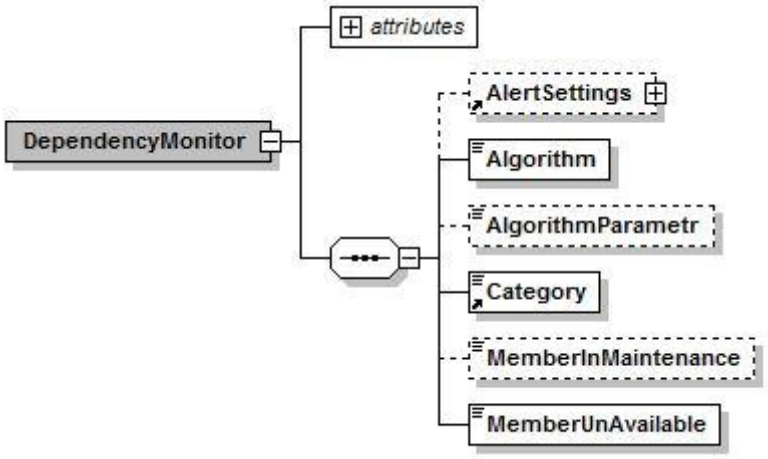

Obrázek 22: Algoritmus "BestOf".

```
<Algorithm>BestOf</Algorithm>
```

Ukázka kódu 19: Nastavení BestOf algoritmu.

### DependencyRollupMonitor

Tento monitor se používá k sestavení modelu zdraví objektů na základě jejich vztahů. Vztahy jsou definovány v mnoha řídících balíčcích. Podobně jako *AggregateRollupMonitor* i *DependencyRollupMonitor* může sledovat skupinu jiných monitorů a v závislosti na nich nastavovat zdravotní stav a také generovat upozornění.




Obrázek 23: Schéma elementu DependencyMonitor.

| Element DependencyMonitor | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Accessibility | Definuje viditelnost elementu z jiných řídících balíčků, nabývá hodnot internal a public (povinný). |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno. (povinný). |
| Target | Cílový objekt (povinný). |
| ParentMonitorID | Monitor, kterému předává svůj zdravotní stav (povinný). |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| Priority | Většinou nastaven na „normal“. |
| RunAs | Zabezpečené reference. |
| RelationshipType | Definuje vazby objektů, nad kterými je následně sestaven model zdraví. |
| MemberMonitor | Monitor z druhé strany vazby, který je zahrnut do modelu zdraví. |

Tabulka 21: Popis atributů elementu DependencyMonitor.

Element category

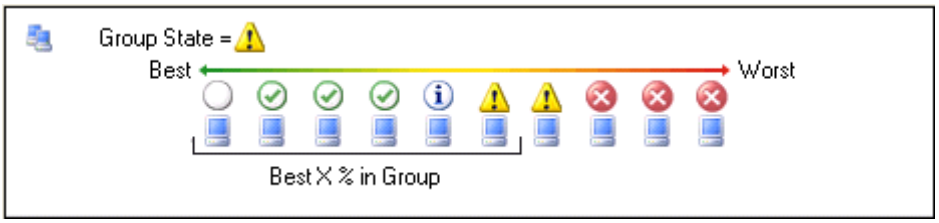
Jako *Category* volíme opět jeden z vysokoúrovňových monitorů (Availability, Configuration, Performance, nebo Security).

Element algorithm

Pro tento element je možné použít algoritmus *WorstOf* a *BestrOf* obdobně jako u *AggregateRollupMonitoru* a navíc je zde algoritmus nazvaný *Percentage*. Tento algoritmus propočítává procento monitorovaných objektů, jež se nacházejí v definovaném stavu.

```
<Algorithm>Percentage</Algorithm>
<AlgorithmParameter>60</AlgorithmParameter>
```

Ukázka kódu 20: Nastavení algoritmu nazvaného Percentage.




Obrázek 24: Procento počítačů v definovaném stavu.

Pokud je zvolen tento způsob vyhodnocování modelu zdraví, je nutné určit hraniční hodnotu. V předchozím příkladu je nastavena na 60 procent.

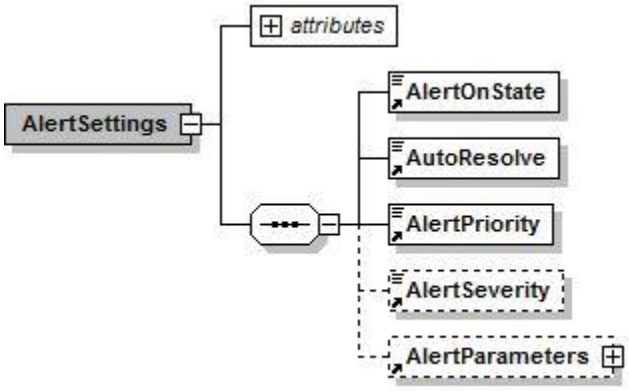
### Elementy memberInMaintenance a memberUnavailable

Elementy specifikují, jak má monitor reagovat, pokud je některý objekt v režimu údržby, respektive nedosažitelný. Oba mohou nabývat následujících hodnot:

- Uninitialized
- Success
- Warning
- Error

### AlertSettings

Tento element obsahují všechny typy monitoru a používá se ke generování výstrah v případě, že se změní model zdraví u sledovaných objektů.




Obrázek 25: Schéma elementu AlertSettings.

*AlertSettings* může definovat pouze jediný atribut *AlertMessage*, který se využívá k lokalizaci a popisu vygenerované výstrahy. Pokud není tato zpráva definována, bude výstraha vygenerovaná se jménem a popisem monitoru, který ji vyvolal.

### AlertOnState element

Tato výstraha je generována, pokud se monitor dostane do námi definovaného stavu. Může to být warning nebo error. Pokud je nejprve vygenerována výstraha pro warning a následně přejde monitor do stavu error, dojde k vygenerování další výstrahy. Vždy je potřeba definovat nejnižší úroveň pro generování výstrahy.

```
<AlertOnState>Warning</AlertOnState>
<AlertOnState>Error</AlertOnState>
```

Ukázka kódu 21: Nastavení elementu OnState na Warning, respektive Error.

### AutoResolve element

Element bez dalších atributů, který slouží k nastavení, zda se má vygenerovaná výstraha automaticky uzavřít, pokud se monitor vrátí zpět do stavu healthy.

```
<AutoResolve >true</AutoResolve >
<AutoResolve>false</AutoResolve>
```

Ukázka kódu 22: Nastavení elementu AutoResolve na true, respektive false.

### AlertPriority

Element, který určuje prioritu výstrahy. Může nabývat hodnot:

- Low
- Normal
- High

```
<AlertPriority>Normal</AlertPriority>
```

Ukázka kódu 23: Definice priority elementu Alert.

### AlertSeverity

Slouží k nastavení úrovně důležitosti dané výstrahy. Může nabývat těchto hodnot:

- Error
- Information
- MatchMonitorHealth
- Warning

```
<AlertSeverity>Information</AlertSeverity>
<AlertSeverity> Error</AlertSeverity>
```

Ukázka kódu 24: Nastavení úrovně výstrahy.

AlertParameters

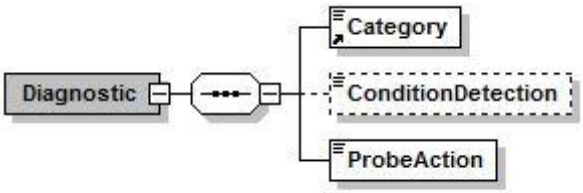
Tento element se využívá k popisu výstrah. Monitor může obsahovat až deset parametrů.

```
<AlertSettingsAlertMessage="Demo.EventBasedMonitor1.AlertMessage">
    <AlertOnState>Warning</AlertOnState>
    <AutoResolve>true</AutoResolve>
    <AlertPriority>Normal</AlertPriority>
    <AlertParameters>
        <AlertParameter1>$Target/Host/Property [Type="Windows!Microsoft.Windows.Computer"]
        /NetworkName$</AlertParameter1>
        <AlertParameter2>$Data/Context/EventNumber$</AlertParameter2>
    </AlertParameters>
</AlertSettings>
```

Ukázka kódu 25: Definice parametrů pro element Alert.

## Diagnostic

Element D*iagnostic* je vázán na konkrétní monitor a může být spuštěn, pokud monitor přejde to určitého zdravotního stavu. Výstup diagnostiky je uložen do databáze a může být později zobrazen v konzoli. *Diagnostic* nemění stav systému, ale pouze jej diagnostikuje. Z toho vyplývá, že definuje element *ProbeAction*. Tento element a jeho atributy byly již popsány v souvislosti s *Task*.




Obrázek 26: Schéma elementu diagnostic.

| Element Diagnostic | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Určuje cílový objekt (povinný). |
| Monitor | Monitor, pro který je diagnostika spuštěná (povinný). |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| ExecuteOnState | Definuje zdravotní stav, při kterém má být diagnostika provedena (Healthy, Warning, Error). |
| TimeOut | Čas, po který smí diagnostika běžet, udaný v sekundách. |

Tabulka 22: Popis atributů elementu Diagnostics.

### Element Category

Hodnota tohoto atributu musí být jedna z následujících možností, jedná se o povinný element.

- AvailabilityHealth
- PerformanceHealth
- ConfiguraitonHealth
- SecurityHealth

Obvykle se volí shodná kategorie jakou má monitor ke které je *Diagnostic* připojen.

### Element ConditionDetection

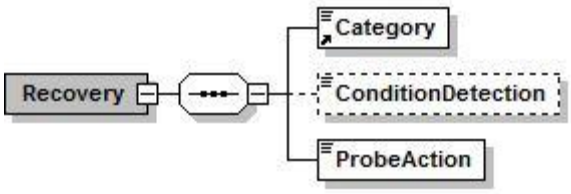
Tento element je volitelný a slouží jako filtr vstupních dat, na základě kterých monitor mění svůj zdravotní stav. Pokud *ConditionDetection* vyfiltruje všechna data, pak se *ProbeAction* neprovede.

| Element ConditionDetection | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| TypeID | ID modulu, ze kterého ConditionDetection dědí (povinný). |

Tabulka 23: Popis atributů elementu ConditionDetection.

## Recovery

Tento element je podobně jako *Diagnostic* vázán na konkrétní monitor a může být spuštěn, pokud monitor přejde to určitého zdravotního stavu. Navíc je možné jej navázat na diagnostiku a spustit jej po jejím ukončení. Výstup *Recovery* je také uložen do databáze a může být později zobrazen v konzoli.




Obrázek 27: Schéma elementu Recoveries.

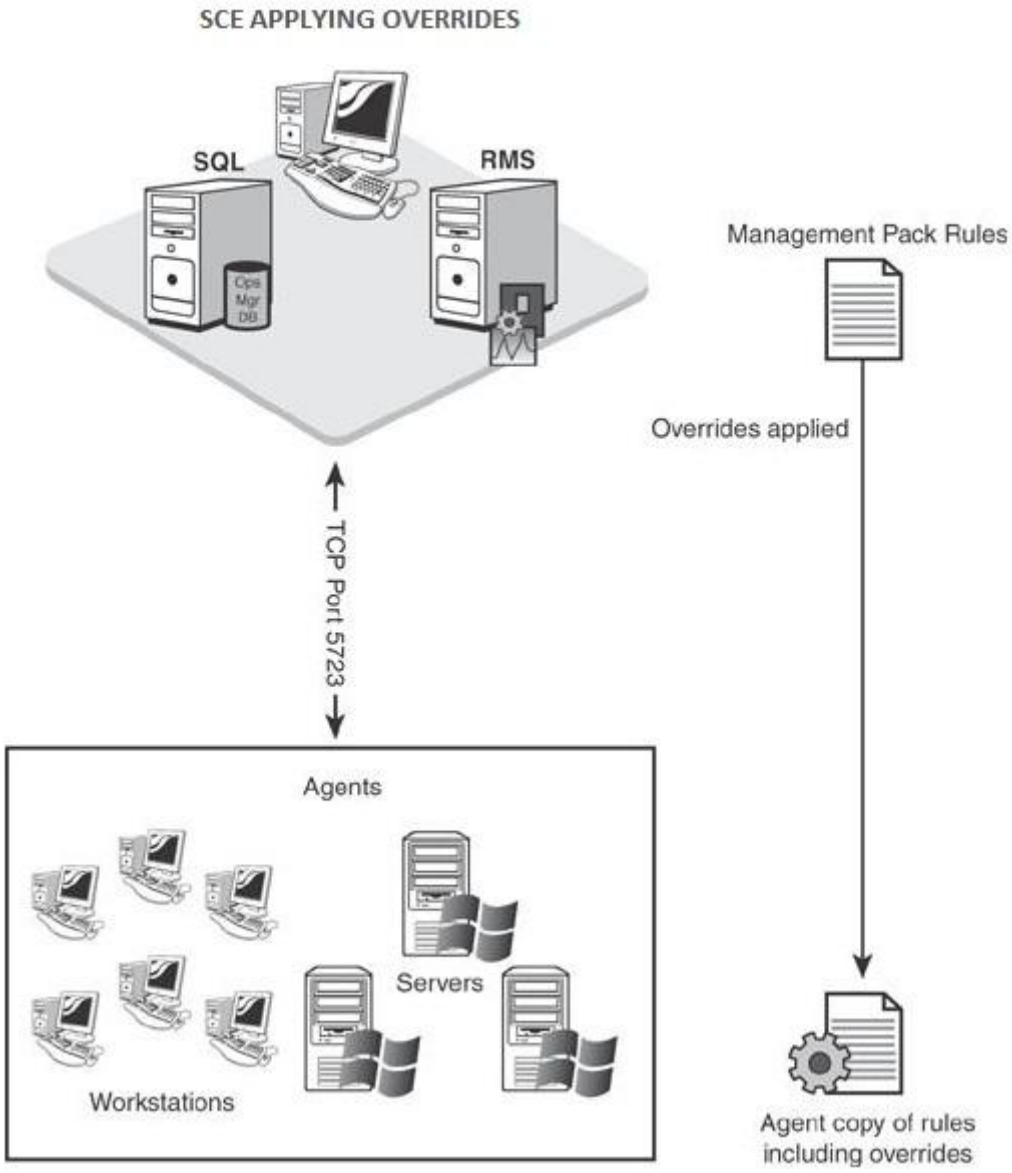
*Recovery* ale na rozdíl od *Diagnostic* běžně mění stav systému. Pokud stav systému není potřeba měnit, je doporučováno použít právě element *Diagnostic*. *Recovery* musí definovat atributy *Category* a *WriteActions* a může obsahovat volitelný atribut *ConditionDetection*. Pro elementy *Category* a *ConditionDetection* platí totéž, co bylo řečeno u *Recovery*. Element *WriteAction* byl popsán v souvislosti s *Rules*.

| **Element Recoveries** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Určuje cílový objekt (povinný). |
| Monitor | Monitor, pro který je diagnostika spuštěná (povinný). |
| ResetMonitor | Definuje, zda se má resetovat monitor (true/false). |
| Remoteable | Definuje, zda bude toto workflow fungovat bez agentů pro monitorování. |
| ExecuteOnState | Definuje zdravotní stav, při kterém má být diagnostika provedena (Healthy, Warning, Error). |
| ExecuteOnDiagnostic | Definuje diagnostiku, po které se recoveries spustí. Pokud definujeme ExecuteOnState, pak již nemůžeme tento atribut použít. |
| TimeOut | Čas, po který smí diagnostika běžet, udaný v sekundách. |

Tabulka 24: Popis atributů elementu recoveries.

## Overrides

Každý řídící balíček je při jeho vydání v základním nastavení definovaném vydavatelem balíčku a popisuje zdravotní stav produkdu, tak jak je definován vydavatelem tohoto balíčku. V případě, že má být balíček využít ve specifickém firemním prostření, je možné k jeho přizpůsobení využít element *Overrides*. Pomocí *Overrides* lze upravit nastavení *Rule* nebo monitoru pro jeden objekt, např. monitorovaný počítač, aniž by bylo nutné jakkoli zasahovat do *Rule* či monitoru. Kupříkladu, pomocí *Overrides* je možné zakázat rule pro jeden konkrétní monitorovaný počítač, aniž by to ovlivnilo funkčnost tohoto rule na ostatní sledovaných objektech. Podobně je možné zvýšit maximální hodnotu využití CPU u vytížených serverů, aby se předešlo falešným výstrahám, a to opět bez ovlivnění ostatních částí monitorovaných systému.

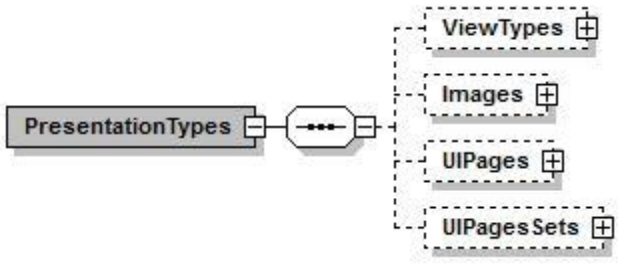



Obrázek 28: Jak SCE aplikuje overrides. Zdroj: Meyler Kerrie, 2008.

*Overrides* můžeme aplikovat na *Monitor*, *Rule*, *Discovery*, *Diagnostics* nebo *Recovery*. Více o jeho elementech a atributech najdeme v literatuře [6] nebo [10].

### 5.7.6 Presentation Types

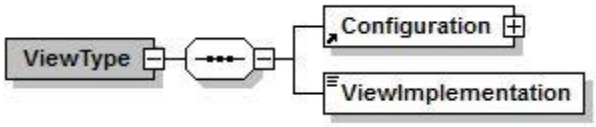
V této sekci řídícího balíčku jsou definovány všechny prvky, které mají souvislost s uživatelským rozhraním.




Obrázek 29: Schéma elementu PresentationTypes.

#### ViewType

ViewType slouží jako šablona pro styl pohledů. Pokud v konzoli vytvoříme nový pohled, můžeme si vybrat z definovaných *ViewType*



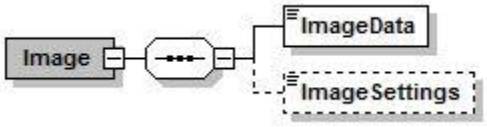

Obrázek 30: Schéma elementu ViewType.

| Element ViewTypes | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Assability | Definuje viditelnost elementu z jiných řídících balíčků, může nabývat hodnot internal nebo public (povinný). |

Tabulka 25: Popis atributů elementu ViewType.

#### Image

Element *Image* obsahuje data, ze kterých je možné sestavit obrázek a informace o něm. Obrázky jsou přiřazeny ke třídám a jiným objektům řídícího balíčku. Pokud je v konzoli zobrazena instance takovéhoto objektu, může být vykreslen obrázek přiřazený k danému objektu.




Obrázek 31: Schéma elementu Images.

| Element Image | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Assability | Definuje, viditelnost elementu z jiných řídících balíčků, může nabývat hodnot internal, nebo public (povinný). |
| Category | Typ obrázku. |

Tabulka 26: Popis atributů elementu Image.

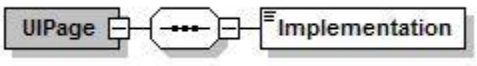
Typ obrázku (category) musí být jeden z následujících typů:

- u16x16Icon
- u32x32Icon
- DiagramIcon
- BmpIcon
- StatusIcon
- BackgroundImage

## UIPage

Element obsahuje reference na Windows Forms, které jsou využity jako průvodce nebo vlastnost stránky.

*UIPage* element vyžívá *IUPageSet* k seskupení stránek.



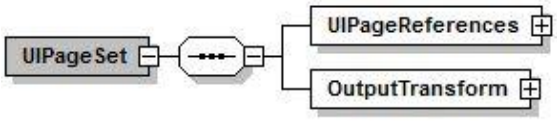

Obrázek 32: Schéma elementu UIPages.

| Element UIPage | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Assability | Definuje viditelnost elementu z jiných řídících balíčků, může nabývat hodnot internal, nebo public (povinný). |

Tabulka 27: Popis atributů elementu UIPage.

### UIPageSet

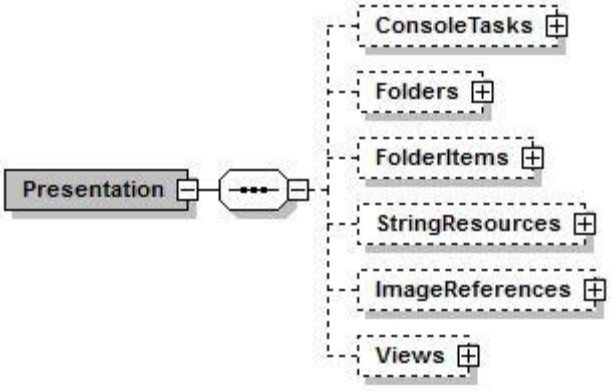
Slouží k uspořádání stránek uživatelského rozhraní do skupin.




Obrázek 33: Schéma elementu UIPageSet.

## 5.7.7 Presentation

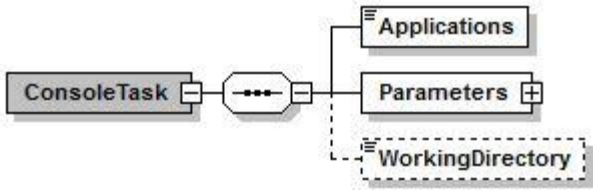
V této sekci řídícího balíčku jsou definovány všechny elementy, které mají vliv na vzhled uživatelského rozhraní.




Obrázek 34: Schéma elemenntu Presentation.

## ConsoleTask

*ConsoleTask* má obdobné vlastnosti, jako již dříve popsaný element *Task* s tím rozdílem, že je spouštěn na základě požadavků z operační konzole.



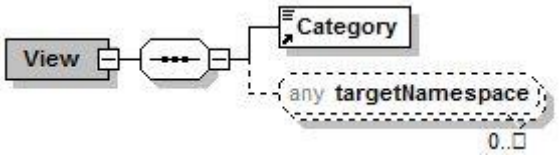

Obrázek 35: Schéma ConsoleTask.

| **Element ConsoleTask** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Cílový objekt (povinný). |
| RequireOutput | Definuje, zda bude pravidlo zapisovat na standardní výstup (povinný). |
| Category | Definuje, zda bude toto workflow spuštěno při monitoringu bez agenta (defaultně true). |

Tabulka 28: Popis atributů elementu ConsoleTask.

## View

Pohledy slouží k zobrazení sesbíraných dat o určitém objektu nebo skupině objektů. Pokud si chceme prohlédnout nějaký pohled, SCE vytvoří dotaz, který odešle na databázi, a výsledek tohoto dotazu je následně zobrazen.




Obrázek 36: Schéma elementu View.

Některé pohledy jsou vytvořeny automaticky při instalaci Essentials, jsou to:

- Active Alerts – zobrazuje všechny výstrahy, které nebyly zavřeny.
- Computers – zobrazuje všechny nalezené počítače.
- Discovered Inventory – zobrazuje všechny nalezené objekty.
- Distributed Applications – zobrazuje všechny objekty vytvořené pomocí designéru distribuovaných aplikací.
- Tasks Status – zobrazuje seznam všech dostupných úloh.

| **Element View** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Comment | Popis autora. |
| Enabled | Definuje, zda je ve výchozím stavu pravidlo povoleno (povinný). |
| Target | Cílový objekt (povinný). |
| TypeID | Určuje ViewType který je zde implementován (povinný). |
| Visible | Definuje, zda daný pohled je viditelný pro uživatele. |

Tabulka 29: Popis atributů elementu View.

Pokud importujeme řídící balíček, je automaticky vytvořená složka v konzoli Monitoring a do této složky jsou uloženy všechny pohledy z importovaného balíčku. Pohledy v této složce se nedají upravovat, ani vytvářet nové. Pokud si přejeme vytvořit vlastní pohled, musíme jej uložit do vlastní složky.

```
<Views>
   <View ID=”ViewID” Comment=”Comment” Accessibility=”Public/Internal”
   Target=”TargetClassID” TypeID=”TypeID” Visible=”True/False”>…</View>
</Views>
```

Ukázka kódu 26: Syntaxe Views.

## Folder

Složky slouží k zařazení pohledů. Každý pohled musí být zařazen do některé ze složek.

| **Element Folder** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní (povinný). |
| Accessability | Definuje viditelnost dané složky. |
| ParentFolder | Definuje nadřazenou složku (povinný). |

Tabulka 30: Popis atributů elementu Folder.

### FolderItem

Tento element slouží k přiřazení jednotlivých pohledů do složek.

| **Element FolderItem** | |
|---|---|
| **Atribut** | **Popis** |
| ElementID | Definuje ID pohledu, který má být zobrazen ve složce s ID definovaném v atributu (povinný). |
| Folder | Definuje ID složky, ve které má být pohled zobrazen (povinný). |

Tabulka 31: Popis atributů elementu FolderItem.

### ImageReferences

Tento element slouží k vytvoření asociací mezi instancemi *ClassType* a obrázky.

| **Element ImageReference** | |
|---|---|
| **Atribut** | **Popis** |
| ElementID | Definuje ClassType pro který je obrázek s tímto ImageID zobrazen (povinný). |
| ImageID | Určuje identitu obrázku, musí být unikátní (povinný). |

Tabulka 32: Popis atributů elementu ImageReference.

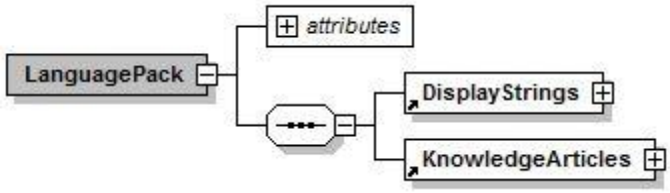
### StringResources

Tento element slouží k definování zprávy, která bude v případě výskytu alertu zobrazena. Tuto zprávu je možné nalézt v sekci language pack pod elementem *DysplayString* v řídícím balíčku. Každý *LanguagePack* musí definovat *DisplayString* element pro všechny definované *StringResource*.

| **Element StringResource** | |
|---|---|
| **Atribut** | **Popis** |
| ID | Určuje identitu elementu, musí být unikátní.<br>Obecně lze říci, že ID končí **.AlertMessage** (povinný). |

Tabulka 33: Popis atributů elementu StringResource.

### 5.7.8 Language Packs




Obrázek 37: Schéma elementu Language Packs.

## DisplayString

Element obsahuje popisky, které se mohou zobrazit v konzoli. Může se jednat například o názvy objektů, případně popisy alertů.

| **Element DisplayString** | |
|---|---|
| **Atribut** | **Popis** |
| ElementID | Určuje identitu elementu, pro který je daný popisek určený (povinný). |
| SubEnabledID | Element, který je povinný pouze v případě, že popisek je určený k subelementu některého z elementu. Obsahuje ID subelementu. |

Tabulka 34: Popis atributů elementu DisplayString.

Tento element definuje dva subelementy *Name* a *Description*. Ty to elementy slouží k zadání jména a popisu *DisplayStringu*.

### KnowledgeArticles

Tento element obsahuje všechny články s informacemi, které se vztahují k danému řídícímu balíčku. Definuje jediný element *KnowlageArticle*.

| **Element KnowledgeArticle** | |
|---|---|
| **Atribut** | **Popis** |
| ElementID | Určuje identitu elementu, pro který je daný článek určený (povinný). |
| Comment | Popis autora. |
| SubEnabledID | Element, který je povinný pouze v případě že článek je určený k subelementu některého z elementu. Obsahuje ID subelementu. |
| Visible | Definuje, zda daný pohled je viditelný pro uživatele. |

Tabulka 35: Popis atributů elementu KnowledgeArticle.

*KnowladgeArticle* může být vytvořen pro jakýkoli objekt řídícího balíčku, ale pouze články spojené s pravidly, které generují alert nebo monitory, mohou být zobrazeny v konzoli SCE.

*KnowlageArticle* se ukládají podle schématu MAML, toto schéma je součástí SCE. Knowlage articles mohou být dvojího typu – MAML, nebo HTML. HTML je zde zahrnuto kvůli zpětné kompatibilitě.

Více k problematice management packu nalezneme v literatuře [7], [8], nebo [11].

### 5.7.9 Přehled objektů řídícího balíčku

Autorizační konzole umožňuje tvorbu základních objektů řídícího balíčku. V některých případech se nevyhneme vytváření objektů v XML kódu nebo alespoň jejich editaci v XML kódu. V následující tabulce je uveden přehled objektů řídícího balíčku a možnosti jejich vytvoření.

| MPs objekt | Kde objekt vytvořit |
|---|---|
| Management Pack file | Autorizační konzole, XML |
| ClassType | Pouze XML |
| Relationship | Pouze XML |
| Object Discovery | Pouze XML |
| Monitor | Autorizační konzole, XML |
| Rule | Autorizační konzole, XML |
| Task | Autorizační konzole, XML |
| Diagnostics | Autorizační konzole (s editací XML), XML |
| Recoveries | Autorizační konzole (s editací XML), XML |
| View | Autorizační konzole, XML |
| Console Task | Pouze XML |
| Folders | Pouze XML |

Tabulka 36: Objekty řídícího balíčku a způsob jak je vytvořit.

# 6 Závěr

Práce se zabývá komplexně problematikou monitorování aplikací. V textové části jsou popsány dva nejčastěji využívané logovací nástroje pro zápis logů na platformě .NET – Log4net a Logging Application Block z Enterprise Library. Dále je detailně popsána struktura řídícího balíčku, pomocí kterého je možné danou aplikaci sledovat. Byla věnována pozornost podrobnějšímu popis jednotlivých součástí řídícího balíčku a jeho tvorbě pomocí jazyku XML. Při pojednávání o logovacích nástrojích byla pozornost zaměřena na možnosti nastavení konfiguračního souboru. Tyto informace, za pomocí ukázek zdrojového kódu v jazyku XML, mohou být využity jako stručný návod pro vytváření vlastního řídícího balíčku v reálné situaci, nelze je však chápat jako kompletní manuál.

V průběhu práce byla vytvořena testovací aplikace, simulující reálnou aplikaci automatického zpracování smyšleného procesu. Tato aplikace je postavena na architektuře klient-server. Komunikace mezi klientem a serverem je zajištěna pomocí frameworku Windows Communication Foundation. V případě serveru je služba WCF hostována v aplikaci postavené na Windows Presentation Foundation. K samotnému zápisu logů je využit Loggingg Application Block z Enetrprise Library, díky kterému je zajištěna možnost snadného nastavení cíle logování nebo jeho formátu. Všechna nastavení jsou velice snadná a přehledná a lze je provádět i za běhu aplikace. Klientská část aplikace je rovněž postavena na technologii WPF.

Další součástí práce je řídící balíček, který prakticky ověřuje možnosti monitorování aplikací automatického zpracování. V průběhu práce bylo vytvořeno několik různých řídících balíčků. Na přiloženém CD se nachází jeden z nich, jenž umožňuje monitorovat logy testovací aplikace a to jak pomocí rules, tak pomocí monitorů. Zajímavý je monitor, pomocí kterého je možné logem s jedním ID překlopit stav monitoru do stavu healthy a následně logem s jiným ID jej překlopit zpět. Díky tomu je možné, aby monitor byl nastaven do původního stavu pomocí aplikace, bez nutností zásahu administrátora v konzoli SCE. Řídící balíček také umožňuje monitorovat zatížení procesoru a paměti serveru a v případě překročení dané meze vygenerovat alert. Krom toho také sleduje počet spuštěných instancí na serveru, kdy vyhovující je právě jedna instance.

System Center Essentials, případně jiný, jemu podobný produkt, je ideálním řešením pro firmy zabývající se dodávkou informačních systému, které chtějí mít neustálý přehled o dění v systému u zákazníka. Díky takovému programu se mohou vyhnout mnoha nepříjemnostem, spojených s chodem aplikace. O vznikajících problémech se mnohdy dozví ještě dřív, než je odhalí zákazník, či v horším případě, než přestane fungovat celý systém.

# Literatura


[1] APACHE. *Logging services* [online]. 9.11.2007 [cit. 2009-6-10]. Dostupné z http://logging.apache.org

[2] MICROSOFT. *Microsoft Enterprise Library* 4.1 [online]. 1.10.2008 [cit. 2009-6-10]. Dostupné z http://msdn.microsoft.com/en-us/library/dd203099.aspx

[3] NEWTON, Keenan. *The Definenitive Guide to the Microsoft Enterprise Library*, United States of America: Apress 2007. 522s, ISBN-10: 1-59059-655-2

[4] LAWSON, Piers. *Get Logging with the Enterprise Library* [online]. 28.9.2005 [cit. 2009-6-10]. Dostupné z http://www.codeproject.com/KB/architecture/GetLogging WithEntLib.aspx

[5] MICROSOFT. *Deployment Guide for System Center Essentials 2010 Beta* [online]. 14.10.2009 [cit. 2010-1-25]. Dostupné z http://www.microsoft.com/downloads/details.aspx?FamilyID=c8f515da-2050-4de2-8129-04ba008c4453&displaylang=en

[6] MICROSOFT. *Operations Guide for System Center Essentials 2010 Beta* [online]. 9.10.2009 [cit. 2010-1-25]. Dostupné z http://www.microsoft.com/downloads/details.aspx?FamilyID=D52F4AA4-D366-4910-9888-4F8C764ACBD3&displaylang=en&displaylang=en

[7] KERRIE, Meyler et al., *System Center Operations Manager 2007 Unleashed*, United States of America: Sams Publishing 2008,1416s ISBN-13: 978-0-672-329555-7

[8] WILSON, Steve, *AuthorMPs* [online]. c2009 [cit. 2010-2-22]. Dostupné z http://www.authormps.com

[9] SILECT. *MP Studio: Simplify the Deployment and Management of Operations Manager* [online]. c2010 [cit. 2010-1-10]. Dostupné z http://www.silect.com/Products/MPStudio.aspx

[10] MICROSOFT. *Systém Center* [online]. c2010 [cit. 2010-1-10]. Dostupné z http://technet.microsoft.com/en-us/systemcenter/default.aspx

[11] MICROSOFT. *Operations Manager 2007 Management Pack Development Kit* [online]. 11.11.2009 [cit. 2010-2-28]. Dostupné z http://msdn.microsoft.com/en-us/library/ee533840.aspx

[12] KOŠEC, Petr. *MS Systém Center Essentials* [online]. 11.10.2007 [cit. 2009-12-12]. Dostupné z http://stream.wug.cz/ms-system-center-essentials/

[13] MILLS, David. *Systém Center Essentials 2007: Overview* [online]. 19.4.2007 [cit. 2010-2-25]. Dostupné z http://www.microsoft.com.tr/sunum/bestofmms/MMS310-Essentials-Overview.pptx

[14] MICROSOFT. *Microsoft's Remote Managed Services Solution for Partners* [online]. 01.07.2006 [cit. 2010-3-28]. Dostupné z http://download.microsoft.com /download/d/c/f/dcfdca55-1316-4ebb-8dd1-901cf73e7533/managed%20services%20 overview(v3).doc

[15] MICROSOFT. *Microsoft .NET Framework* [online]. c2009 [cit. 2010-4-28]. Dostupné z http://www.microsoft.com/net/

## Seznam tabulek

## Seznam obrázků

## Seznam ukázek kódu